EXHIBIT A

Don Turnbull, Ph.D.

donturn@gmail.com
512.428.8763

Overview
Highly effective software developer, designer and researcher with over 20 years experience. Contributor to multiple commercially successful consumer and enterprise applications including Content Management Systems, CASE tools, desktop utilities, search tools and ecommerce Web sites. Accomplished researcher and creator of innovative, patented and trade-secreted technologies related to information retrieval, behavioral modeling, content organization and analytics. Author of numerous academic publications including: a book on Web-based information seeking and knowledge work, articles on human-computer interaction design, personalization for information retrieval and recommender systems, as well as numerous definitive works on information architecture methodologies, designs and implementations.

Principal – InfoTheory/Don Turnbull, ULC **2002-ongoing**
Advising software companies, design agencies and information services corporations on the research and development of systems architectures, data science activities and human-computer interaction. Directing the design of multi-platform applications (desktop and mobile) for consumer-oriented information systems including Web, smartphone and tablet interfaces, information architectures and ecommerce recommendation systems as well as advising on future development based on data-centric architectures and for re-designing existing systems in use.

Analyzing intellectual property, patent portfolios and innovative technologies to author reports, research software architecture and software development methods, create new IP, advise intellectual asset development, as well as serve as an expert in patent-related cases with prior art, infringement or validation issues.

Designed and prototyped information retrieval systems and application programming interfaces for consumer and enterprise search systems involving indexing, tagging and faceted-metadata methods. Created techniques and organization schemes to instrument systems for collection and analysis of empirical behavioral data logs from Web site usage and user-generated content. Developed data models and interaction strategies for consumer and vertical channel mobile devices for information retrieval, storage and management.

Advised and coordinated a very large, multiple partner contract systems development effort as the technical architect for a back-office ecommerce enterprise portal that included vendor tool selection, managing and negotiating among partners for project management and high-level goal definition. Devised and configured analytics tools as well as acting upon analysis findings to provide guidelines for site information architecture and user experience designs.

Designed and directed creation of a knowledge management system to provide efficient development workflow, search functionality and knowledge discovery from intranet information sources including the design and deployment of enterprise wikis, blogs, social networking and workgroup collaboration tools, source code version control and office documents including (OLAP) reporting and financial applications. Lead efforts to build upon open source applications and protocols including novel interface designs, autonomous agents and collaborative filtering to improve information access and use in the organization.

Advisor – ThinkCX **2016 - present**
Directing research efforts for large-scale machine learning platform for collecting, measuring and predicting user behavior across mobile device and geo-locational activity traces. Advised on strategic use and extension for a social media analysis framework for user behavior prediction. Advising on the startup process, general technology and inventing as well as authoring patents.

Advisor – University of Texas at Austin Technology Incubator 2009-2011 & 2013 - 2018
Mentor and Advisor with the University of Texas at Austin, Austin Technology Incubator and IC^2 (Innovation, Creativity and Capital) Institute to engage local and international technology companies (primarily software startups) in advancing research, providing strategic expertise, intellectual property evaluation, market assessment and assisting in designing products for market.

Research Computer Scientist – Tapstream 2013
Researched and invented novel systems and designs in desktop and mobile computing, data science, user modeling and analytics domains to produce intellectual property for existing and future technologies.

Investigated the state of the art and the competitive landscape for software services and advising directions for future product development.

Principal Data Scientist & Architect – Wyley Interactive 2012-2013
Researched and developed algorithms, data sets and software architectures for a mobile games discovery, recommendation and rewards system. Created intellectual property including system designs, algorithms and computational methods for patent-pending systems. Wrote and coordinated research grants and other funding programs.

Architected an empirical business intelligence analytics platform for understanding and predicting user acquisition, monetization strategy, app distribution, gameplay telemetry, operation costs and social interaction in the mobile gaming space.

Assistant Professor – University of Texas at Austin 2002-2009
Created and taught graduate-level courses and development labs in Information Architecture, Interaction Design & Human Computer Interaction (HCI), Web Analytics, Web Information Retrieval Evaluation & Design (search), the Semantic Web and Knowledge Management systems. Investigated very large-scale data mining systems and algorithms (including Web use data for personalization), interface designs for multimedia access and Web search engines. Co-Principal Investigator for Web content classification and collaborative filtering system (the OpenChoice Project) including system architecture, algorithm evaluation, interface design and user coordination. Conducted and collaborated on Information Retrieval system development for blog analysis and topic distillation tasks including spam detection and initial sentiment analysis.

Explored search engine technologies (multimedia, indexing, interaction), search engine optimization (natural organic search, personalized search, sponsored advertising search) as well as creation and empirical analysis of behavioral model of search user experience towards improving the search process.

Advised graduate students and managed research team efforts for information technology research and development including Semantic Web applications, mobile device interfaces, Content Management Systems, Web browser software analysis, Web accessibility evaluation, Web link mining and analytics, information architecture design methodologies, and Web advertising plans and tools.

Director of Advanced Development − Outride, Inc. (acquired by Google, Inc.) 2000-2001
Created and Coordinated intellectual property assets including patent applications and licensed patents from Xerox PARC as well as original work developed at Outride. Authored patents relating to personal relevance models for information retrieval, information privacy and e-commerce systems in networked and mobile environments. Worked with attorneys to manage, track, develop and revise patent portfolio. Authored several trade-secreted technologies, patent applications and at least one patent (7,089,237) for interfaces and systems that display content for commerce activities in mobile and/or networked environments which could include desktop, smartphone, tablet or set top devices.

Initiated and managed Competitive Intelligence efforts to scan for emerging technologies including research reviews, attending conferences and analyzing competitor technologies. Organized intelligence resources for engineering and legal purposes to protect and augment existing intellectual property. Maintained the Competitive Intelligence database as a knowledge management activity and served as a technical strategic advisor for business partners.

Acted as research advisor for all corporate data mining, interface designs and usability studies activities. Worked closely with User Interface design team to design product specifications for an application to search the Web; manage bookmarks; view and search Web use history; and interact with a directory of Web-based resources. Designed high-level technical architecture and interface for a Web browsing privacy application to enable users to control and edit data collected about their Web use activities.

Managed Metrics project with vendors to provide a value proposition for Outride technology for business development. Selected an external testing agency; designed the initial tests; determined evaluation criteria; selected competing technologies; designed data collection methods; analyzed test data; and edited the final report. This extremely successful project served as a key asset in demonstrating Outride technology to investors, business partners and industry analysts and additionally used extensively in press releases and corporate product literature.

Research Scientist & WebTracker Development Lead − University of Toronto 1997-2000
Planned and implemented a 16-month study to develop a comprehensive understanding of corporate Internet use utilizing a synthesis of data collection and analysis methods including an initial survey questionnaire; software to collect use data gathered with a custom-developed Web tracking application; and interviews with study participants.

Analyzed data using qualitative and quantitative methods to test hypotheses of new models of Information Seeking and Information Retrieval behavior. Used study results to make recommendations on improving organizational Knowledge Management and individual Web use techniques, as well as to design new software tools to coordinate and leverage organizations' intranet and Internet use.

Designed and prototyped WebTracker: a client-side data collection instrument for transparently logging Web browser use. Researched data collection methods, instruments for Internet protocols and network-enabled client applications. Automated the data mining of WebTracker logs with customized analysis tools to build both individual and aggregate models of Web use. Initiated consortium with other research institutes to expand WebTracker use.

Technical Architect: Internet Applications − IBM Interactive Media Group 1996
Designed and authored specifications for hybrid (CD-ROM and Internet) World Book-IBM Interactive Multimedia Encyclopedia involving data formats, user interface, indexing structures and versioning controls. Researched and co-developed patented the TRUE/IP protocol for registering, updating and exchanging client-server information via the Internet. Prototyped large-scale collection and analysis of client application and Internet use data.

Managed technical staff in product manager role with multiple vendors and locations including interviewing, hiring, training and planning state-of-the-art development labs. Developed technical architecture for e-Business services using database technology for user profiling to enable content personalization. Directed in-house usability efforts and commercial opportunities for existing technologies. Prototyped Web site building service included with all IBM Small Business System sales.

Knowledge Management Researcher − AT&T 1995
Designed and constructed ISO 9000-compliant Web-based Knowledge Management system for corporate technical information. Researched and developed an iterative methodology to develop, organize and publish interactive documents using object-oriented content classification and user-centered design principles. Trained technical staff in this new methodology including coursework and system templates.

Sr. Information Developer − MicroHelp, Inc. 1994-1995
Programmed file Windows utility software application including file metadata analysis and duplication detection. Designed user interfaces including metaphor creation, dialog layout and icon design for *UnInstaller for Windows*. (In 1994, the best-selling utility in the United States- 4 million copies sold.) Conducted usability studies including designing test scenarios; user modeling; monitoring and recording test data; and analyzing resulting data. Developed programs to automatically generate hypertext documentation from print documentation. Designed and programmed interactive multimedia applications to demonstrate software products.

Technical Editor − Macmillan/SAMS Publishing **1994**

Edited object-based visual programming and software development books for technical accuracy, initially for ObjectView, a product I designed for KnowledgeWare. Wrote and tested programming examples and database overview chapters used in various publications.

Methodologist − KnowledgeWare, Inc. **1991-1994**

Managed project team through development cycle of Computer-Aided Software Engineering (CASE) tools including finalizing requirements, organizing development team, running status meetings, reviewing documentation, testing, prioritizing development issues and designing future enhancements. Used industry standard software engineering methodologies and frameworks (including Information Engineering and Rapid Application Development) for large-scale software projects.

Researched software engineering methodologies to design methods and technologies for next generation CASE tools. Implemented designs included an object-based interface builder and large-scale hypertext information authoring and content management applications using graphical objects, with SGML (GML) formatting and semantics, WYSIWYG editing as well as link management. Designed and reviewed all graphical user interfaces for compliance. Created all corporate usability and interface design standards including task analysis methods to improve products.

Education

Ph.D. Information Studies - University of Toronto, 2002

Dissertation: "Knowledge Discovery in Databases of Web Use: A Search for Informetric and Behavioral Models of Web Information Seeking"

M.S. Information, Design & Technology - Georgia Institute of Technology, 1995

Thesis: "Object-Oriented Information Development: A Methodology and System for Large-Scale Hypertext Documents" (Web server design and Semantic Web content organization and deployment)

B.A. General Studies - University of Texas at Arlington, 1988

Knowledge Engineering (Computer Science & Cognitive Science)

Books

Choo, Chun Wei, Brian Detlor, and Don Turnbull. (2000) Web Work: Information Seeking and Knowledge Work on the World Wide Web. The Netherlands: Kluwer Academic Publishers.

Book Chapters

Dillon, A., & Turnbull, D. (2010) Information Architecture. Encyclopedia of Library and Information Science, 2010, (3rd Ed.). Taylor & Francis.

Dillon, A., & Turnbull, D. (2006) Information Architecture. Encyclopedia of Library and Information Science, 2006. Taylor & Francis.

Turnbull, D. (2005). World Wide Web Information Seeking. In K. E. Fisher, S. Erdelez (Eds.), Theories of Information Behavior. Medford, New Jersey: Information Today, Inc.

Selected Journal Articles

Turnbull, D., & Bright, L. F. (2008). Advertising Academia with Sponsored Search: An Exploratory Study Examining the Effectiveness of Google AdWords at the Local and Global Level. International Journal of Electronic Business, 6(2), 149-171.

Pitkow, J., Schutze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., et al. (2002). Personalized Search: A Contextual Computing Approach May Prove a Breakthrough in Personalized Search Efficiency. Communications of the ACM, 45(9), 50-55.

Edmonds, K. A. A., Bluestein, J. J., & Turnbull, D. (2006). A Personal Information and Knowledge Infrastructure Integrator. Journal of Digital Information, 5(1).

Choo, C. W., Detlor, B., & Turnbull, D. (2000). Information Seeking on the Web: An Integrated Model of Browsing and Searching. First Monday, 5(2).

Selected Conference Papers (refereed)

Turnbull, D. (2007). Rating, Voting & Ranking: Designing for Collaboration & Consensus. Paper presented at the Association of Computing Machinery Computer Human Interface Conference (SIGCHI), San Jose, CA.

Turnbull, D. (2006, May 23, 2006). Methodologies for Understanding Web Use with Logging in Context. Paper presented at the The 15th International World Wide Web Conference, Edinburgh, Scotland.

Dillon, A., Kleinman, L., Bias, R., Choi, G. O., & Turnbull, D. (2004). Reading and Searching Digital Documents: An Experimental Analysis of the Effects of Image Quality on User Performance and Perceived Effort. Paper presented at the American Society of Information Science and Technology Annual Meeting, 2004. Information Today, 267-273.

Choo, C. W., Detlor, B., & Turnbull, D. (2000). Working the Web: An Empirical Model of Web Use. Paper presented at the 33rd Hawaii Intl. Conference on System Science (HICSS), Maui, HI.

Turnbull, D. (1999). Interacting with Recommender Systems. Paper presented at the ACM SIGCHI (Computer-Human Interface) Workshop on Recommender Systems, Pittsburgh, PA.

Choo, C. W., Detlor, B., & Turnbull, D. (1999). Information Seeking on the Web - An Integrated Model of Browsing and Searching. Paper presented at the Proceedings of the 62nd Annual Meeting of the American Society of Information Science, Washington, D.C.

Choo, C. W., Detlor, B., & Turnbull, D. (1998). A Behavioral Model of Information Seeking on the Web - Preliminary Results of a Study of How Managers and IT Specialists Use the Web. Paper presented at the Proceedings of the 61st Annual Meeting of the American Society of Information Science, Pittsburgh, PA.

Selected Conference Presentations, Panels & Posters (refereed)

Turnbull, D. (2010). Quantitative Information Architecture. Presented at the American Society of Information Science & Technology Information Architecture Summit, Phoenix, AZ.

Turnbull, D. & Tolva, J. (2010). Metropolitan Information Architecture. Presented at the American Society of Information Science & Technology Information Architecture Summit, Phoenix, AZ.

Turnbull, D. (2009) Information Technology Diversity: Disruptive Technologies, Innovation & Management. Presented at the American Society of Information Science and Technology Annual Meeting, Vancouver, British Columbia, Canada.

Turnbull, D. (2009) Behavioral Checklist for Information Architecture. at the American Society of Information Science & Technology Information Architecture Summit, Memphis, TN.

Turnbull, D. & Bright, L.F. (2008) Advertising & Awareness with Sponsored Search: an exploratory

study examining the effectiveness of Google AdWords at the local and global level. Presented at the American Society of Information Science and Technology Annual Meeting, Columbus, OH.

Turnbull, D., Mackenzie, M., & Edmonds, A. (2007). eManagement in the Dynamic Digital Environment. Paper presented at the World Congress on Management of eBusiness, Toronto, Ontario.

Turnbull, D. (2007). Hide and Seek: the Information Architecture & Design for working with filtered content in the OpenChoice filtering project. Presented at the American Society of Information Science & Technology Information Architecture Summit, Las Vegas, NV.

Turnbull, D., Campbell, D. G., & Fast, K. V. (2007). The Grand Challenges in Information Architecture. Presented at the American Society of Information Science & Technology Information Architecture Summit, Las Vegas, NV.

Turnbull, D. (2006). Setting the Agenda for IA Research. Presented at the Information Architecture Summit, Vancouver, BC Canada.

Turnbull, D., & Efron, M. (2006). OpenChoice: A Platform for Web Content Classification & Filtering. Paper presented at the 15th International World Wide Web Conference Open Source Workshop, Edinburgh, Scotland.

Turnbull, D., Dillon, A., Morville, P., Kaplan, N., Froehlich, T. J., & Robins, D. (2005). The Process of Curriculum Development for Information Architecture. Presented at the American Society of Information Science & Technology Information Architecture Summit, Montreal, Quebec, Canada.

Turnbull, D., Detlor, B., Mackenzie, M., & Edgar, B. (2005). How technology can move in concert with organizational change. Presented at the American Society of Information Science and Technology Annual Meeting, Charlotte, NC.

Turnbull, D. (2004). XIA: Xtreme Information Architecture. Paper presented at the American Society of Information Science & Technology Information Architecture Summit, Austin, TX.

Jobst, J., & Turnbull, D. (2004). Joint Evolution of Web Browsers and Online Information Architecture. Presented at the American Society of Information Science & Technology Information Architecture Summit, Austin, TX.

Burkart, J., Turnbull, D., Vigil, A., Switzky, A., Miranda, D., & Liaw, L. (2004). XIA@UT: An Extreme Makeover. American Society of Information Science & Technology Information Architecture Summit. Feb. 28, 2004. Paper presented at the American Society of Information Science & Technology Information Architecture Summit, Austin, TX.

Turnbull, D. (1998). Data Mining Web Use: Discovering Patterns and Models of Web Information Seeking Behavior using WebTracker Software Application. Paper presented at the IBM Center for Advanced Studies Conference '98 (CASCON 98), Toronto, ON.

Selected Conference Workshops (refereed)

Turnbull, D. (2006, November 4) The Effects of Information Overload on Information Seeking & Use. Workshop paper for the American Society of Information Science and Technology Annual Meeting, SIG USE workshop. Austin, TX.

Turnbull, D. (2006, May 23) OpenChoice: A Platform for Web Content Classification & Filtering. Open Source Workshop at the 15th International World Wide Web Conference. Edinburgh, Scotland.

Turnbull, D. (2003, Oct 18) New Approaches for Studying and Building Information Seeking Models: A Possible Hybrid Approach. SIGUSE Workshop on Information Seeking theory for the American Society of Information Science and Technology (ASIST) 2003 Annual Meeting, Long Beach, CA.

Selected Invited Talks & Panels

Turnbull, D. (2011, October 31). New Directions in Taxonomy (Conference Keynote). Presented at

Taxonomy Boot Camp/KMWorld Enterprise Search Summit, Washington D.C.

Turnbull, D. (2011, March 12). Left Brain Search = Google, Right Brain Search = X? Presented at the South by SouthWest Interactive, Austin, TX.

Turnbull, D. (2010, July 13). Quantitative UX. Presented at the Toronto UX Irregulars. Toronto, ON.

Turnbull, D. (2010, June 28). Semantic Discovery: Making Search Better. Presented at Semantic Web Vancouver. Vancouver, BC.

Turnbull, D. (2010, June 22). Quantitative Information Architecture. Presented at the Vancouver User Experience Group. Vancouver, BC.

Turnbull, D. (2009). Agile UX. Presented at IxDA Austin, Austin, TX.

Turnbull, D. (2008, February 21). Making Search Better in Interactive Media. Presented at the New University of Lisbon (FCSH/UNL). Lisbon, Portugal.

Turnbull, D. (2008, February 26). Making Search Better with Personalization and Informetrics. Presented at the Faculty of Engineering, University of Porto (FEUP). Porto, Portugal.

Turnbull, D. (2006, March 12). Tagging 2.0. Presented at the South by SouthWest Interactive, Austin, TX.

Turnbull, D. (2005, March 13). Leveraging and Augmenting Solipsism. Presented at South by SouthWest Interactive, Austin, TX.

Turnbull, D. (2004, March 14). Revolutionary Search Technologies. Presented at South by SouthWest Interactive, Austin, TX.

Turnbull, D. (2004, March 11). Designing for Pervasive Computing. Austin Mobility Roundtable, the Third Annual Meeting of the Global Mobility Roundtable, Austin, TX.

Turnbull, D. (1998). Leveraging Web Use for Business Intelligence: Discovering Patterns and Classes of WWW Users. IBM Centre for Advanced Studies Research Forum, Toronto, ON.

Other Publications

Bagheri, E. & Cheung J. (Eds) & Turnbull, D. (Industry Track). (2018) 31st Canadian Conference on Artificial Intelligence. Springer Intl Publishing Subseries on Lecture Notes in Computer Science.

Turnbull, D., Jansen, B. J., Hawkey, K., Kellar, M., & Edmonds, K. A. A. (2007). Introduction to Logging Traces of Web Activity special issue. IEEE Journal of Web Engineering, 6(3), 193-195.

Turnbull, D. (1998). Data Mining Web Use: Discovering Patterns and Models of Web Information Seeking Behavior using WebTracker Software Application. Paper presented at the IBM Center for Advanced Studies Conference ‘98 (CASCON 98), Toronto, ON.

Professional Associations

- Association for Computing Machinery (ACM)
- Association for the Advancement of Artificial Intelligence (AAAI)
- Canadian Artificial Intelligence Association
- Python Software Foundation
- American Society of Information Science & Technology (ASIST)
- World Wide Web Conference - Browsers and UI Program Committee
- Information Architecture Institute Advisory Board Member
- EFF-Austin Board of Directors & Advisory Committee
- World Wide Web Consortium - Web Characterization Activity

Fellowships, Scholarships and Grants

- University of Texas Dean’s Fellowships 2003, 2006, 2007 & 2008
- Co-Principal Investigator - OpenChoice classifier & social filtering engine - IMLS 2005-2007

- Temple Foundation Fellowship 2004 & 2005
- University of Texas at Austin Teaching Fellowship 2004
- Google Search Appliance Grant 2003
- University of Texas at Austin John P. Common Teaching Fellowship 2003
- Microsoft ClearType Research Grant 2003
- University of Toronto Open Fellowship 1997, 1998 & 1999.
- Mary H. Beatty Fellowship 1996

Patents and Patent Applications

Turnbull, Don & Muxworthy, Derek & Nielsen Aaron David (applied 2018). System and Method For Measuring And Predicting User Behavior Indicating Satisfaction And Churn Probability. US Patent Application 16/288,014. Assignee: ThinkCX

Turnbull, Don & LaPierre, Larry (applied 2012). Method to Deconstruct Computer Games into Alternative Slices and System to Enable Event Definitions, Achievements and Rewards via a Central Application or Interface Platform. US Provisional Patent # 61/679,605. Assignee: Wyley Interactive.

Turnbull, Don & Schuetze, Hinrich (applied 2001, granted 2006). Interface and system for providing persistent contextual relevance for commerce activities in a networked environment. US Patent # 7,089,237. Assignee: Google, Inc.

Cases involving In-Court Appearance, Testimony or Deposition (retaining party in **bold**)

1. Speedtrack, Inc. v. Wal-Mart Stores, Inc. and **Endeca Technologies, Inc.**, et al. U.S. District Court for the Northern District of California, Case No. 4:06-cv-07336-PJH.
2. **Bid for Position, LLC** v. AOL, LLC, et al., U.S. District Court for the Eastern District of Virginia, Case No. 2:07-cv-00582-JBF-TEM.
3. SFA Systems, LLC v. **Amazon.com, Inc**., et al. U.S. District Court for the Eastern District of Texas Tyler Division, Case 6:11-cv-00052-LED.
4. Softview LLC v. **Motorola Mobility Inc**., et al. U.S. District Court for the District of Delaware, Case No. 1:10-cv-00389-LPS.
5. Rotatable v. **Rackspace, US, Inc., et al.** U.S. District Court for the Eastern District of Texas Marshall Division and Inter Partes Review, USPTO PTAB. Docket: 47015.115.
6. Global Sessions LP and Global Sessions Holdings SRL v. **Comerica, TD, et al.** U.S. District Court for the Western District of Texas, Case Nos.: 1:13-CV-688-SS; 1:13-CV-692-SS, 1:13-CV-691-SS.
7. Enterprise Systems Technologies, S.A.R.L. v. **Apple Inc.**, to the USPTO PTAB. Civil Action Nos.: 14-cv-765-LPS.
8. Enterprise Systems Technologies, S.a.r.l. v. **Microsoft** (**HTC** & **Google** as interveners). International Trade Comission Case No. 337-TA-925.
9. Queen's University at Kingston v. **Samsung Electronics Co, LTD**. Inter Partes Review, USPTO PTAB Case No IPR2015-00583.
10. **Express Mobile, Inc.** v. BigCommerce, U.S District Court for the Easter District of Texas Marshall Division, Case No. 2:17-cv-00130.
11. **Express Mobile, Inc.** v. eGrove Systems Corporation, U.S. District Court of Delaware, Case No. 1:17-cv-703-RGA.
12. Seven Networks, LLC. v. **Google, LLC**, U.S. District Court for the Eastern District of Texas Marshall Division, Case No. 2:17-CV-442-JRG

EXHIBIT B

How to Do *Everything* with Your Zire™ Handheld

Get the most from your Zire handheld

Take notes, schedule appointments, and manage your address book

View Word documents, read e-books, play games, and more!

Dave Johnson
Rick Broida

McGraw Hill OSBORNE

How to Do *Everything* with Your Zire Handheld

Dave Johnson
Rick Broida

McGraw-Hill/Osborne

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

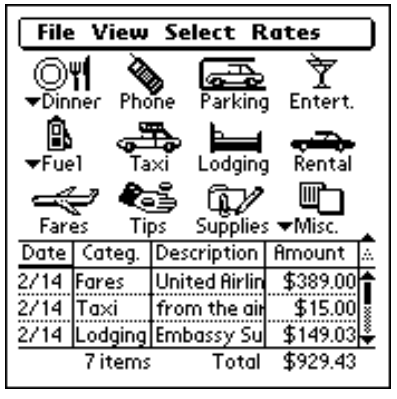
0-07-223047-9

The material in this eBook also appears in the print version of this title: 0-07-222930-6.

DOI: 10.1036/0072230479

ExpensePlus

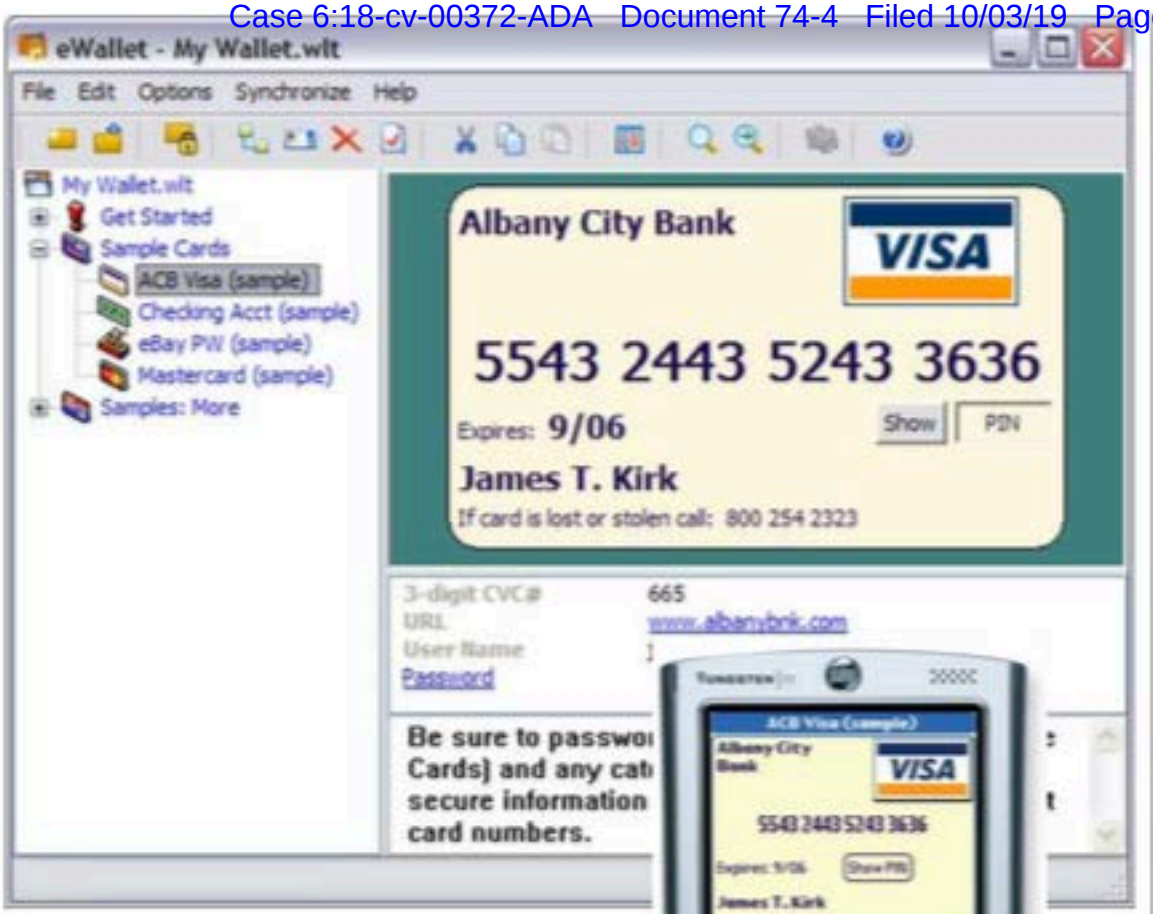
One of the most robust and versatile expense managers available, WalletWare's *ExpensePlus* uses an icon-based interface to simplify the selection of expense types and automation to fill in dates and amounts for things like hotel stays and car rentals. More important, it can link directly to any existing company expense forms created in Excel or FileMaker (including the Mac versions), so you needn't contend with nonstandard forms. Also, if your company's forms aren't based in Excel, WalletWare can design a custom link (for a fee) to other software programs.



Where to Find It

Web Site	Address	What's There
Handmark	**www.handmark.com**	MobileSafe
Chapura	**www.chapura.com**	Cloak
Tranzoa	**www.tranzoa.com**	OnlyMe
CIC	**www.cic.com**	Sign-On
Infinity Softworks	**www.infinitysw.com**	FC Plus Professional, powerOne series
Synergy Solutions	**www.synsolutions.com**	SynCalc
Iambic Software	**www.iambic.com**	ExpenseDirector
WalletWare	**www.walletware.com**	ExpensePlus

EXHIBIT C



Ilium Software eWallet®

Users Guide and Reference

for Windows PCs and Palm Powered™ handhelds

Version 4.0

Ilium Software

3759 Prospect
Ann Arbor, MI 48105
USA
http://www.iliumsoft.com
info@iliumsoft.com
tel: +1 734 973 9388
US tollfree: 888 632 5388
fax: +1 734 973 2640

ilium SOFTWARE®

Simple Software for a Simpler Life

EXHIBIT D

A Personalized Offer Presentation Scheme for Retail In-Store Applications

Yew-Huey Liu, Jih-Shyr Yih, and Trieu C. Chieu

IBM T. J. Watson Research Center,
P.O.Box 704,
Hawthorne, NY 10532
{yhliu,jyih,tchieu}@us.ibm.com

Abstract. Retailers are constantly seeking new innovations to improve people's shopping experience in order to deliver greater consumer and business values. One key objective is to keep the shoppers internet-connected for seamless and informed shopping, and be offered with timely and relevant shopping ideas. Complementing the existing Point-Of-Sale (POS) system, a new retail in-store server supporting personal mobile devices or kiosks is emerging in the retail chains towards the objective. This paper introduces such an in-store server and its role in the overall retail architecture, with the main focus placed on the in-store offer presentation scheme for personalizable service. A lightweight keyword-based rule engine is proposed for selecting offers. A detailed rule processing flow and an efficient implementation for the engine are described. For the ease of reviewing presented offers on a limited display space of a wireless shopping device, an offers layout method which organizes presented offers with individual items is also suggested. The in-store server can lead to seamless multi-channel collaborative shopping.

1 Introduction

Since 1995, we have seen the explosion of B2C electronic commerce. For an enduring web store, exemplified and excelled by Amazon, it goes beyond the usual catalog browsing and order entry, and generates "stickiness" of the site by adding rich information such as latest sales, personalized promotions and consumer tips. However, despite the innovations, the percentage of retail GDP contributed by web-hosted e-commerce is still projected to be in the lower single digits for this decade. The majority of the retail commerce still takes place inside the "brick-and-mortar" branches, where there are plenty of opportunities in innovating as the in-store channel and in collaborating with other channels.

It is a common practice among retailers to use loyalty schemes to encourage purchases by offering points or cash rewards. Retailers also get to collect individual purchase history via such loyalty schemes. However, shoppers are usually anonymous in the store until they reach the checkout counter, where they identify themselves as loyalty customer, and then were offered coupons for future use. Statistics show that a low percentage of such coupons issued in this way are ever redeemed, reflecting the inconvenience of collecting and carrying paper coupons and the low degree of relevance to many shoppers. Additionally, there is no chance to influence or assist the customers' in-store shopping in real-time, other than with generic point-of-promotion displays, which are non-personalized and will be irrelevant to most shoppers.

K. Bauknecht, M. Bichler, and B. Pröll (Eds.): EC-Web 2004, LNCS 3182, pp. 296–304, 2004.

To catch up with shopper expectation, a small subset of retailers are actively engaging in testing a variety of advanced in-store techniques with a new type of in-store commerce server [1]. Figure 1 shows an example of such server introduced by IBM. This in-store commerce server complements the existing POS system and supports a variety of wireless personal mobile devices and kiosks. The in-store server is also used as an integration platform to synchronize information with other channels and enterprise applications, for the delivery of real-time product information, personalized promotions and consistent shopping experience to shoppers at every touch point. The new server investment is supposed to be offset by shoppers' additional purchases through the personalized promotions.

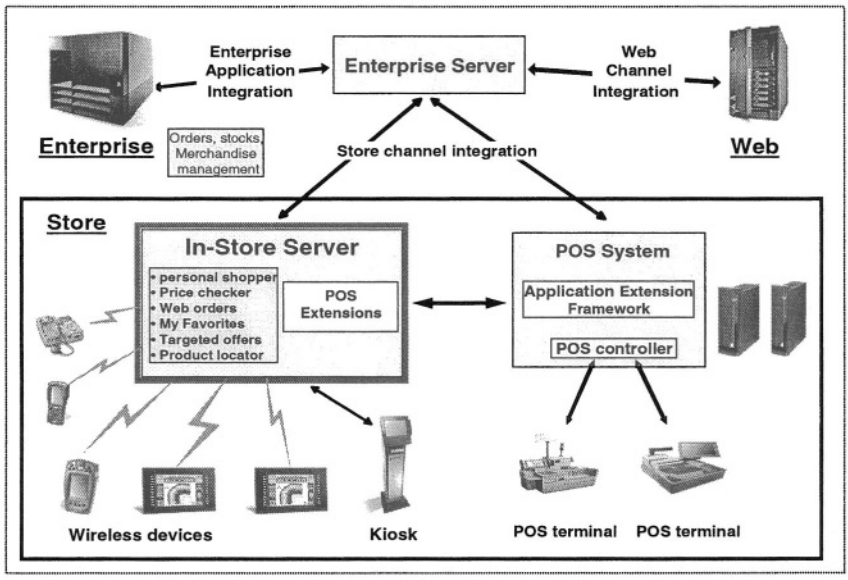


Fig. 1. In-Store Server in a Multi-Channel Environment

Delivery of personalized promotions to shoppers is not a new concept. *Recommender* systems [2] utilizing predicate-based rules, derived from historical or demographic information corresponding to shopper groups, have been deployed in enterprise web sites. There are a number of special considerations that a retailer needs to consider in constructing a recommender system for in-store offer presentation. For examples: 1) A store branch does not have a dedicated IT department for server operations and database maintenance, and can only afford a low-end, personal-computer based in-store server, which has to handle about 200 concurrent shopping sessions. The in-store offer presentation engine needs to be light weight and in small footprint, to meet the CPU and memory usage requirements. 2) There are unique in-store only characteristics such as shopper location and movement within the store. 3) The anticipated limited display size of wireless shopping devices also requires special design considerations for offer layout scheme. 4) Finally, there are always ease-of-use challenges to the programming of offer presentation rules by the retail operators, who don't want to deal with over complex languages.

To address these issues, this paper describes a practical implementation of an intelligent in-store server with an efficient keyword-based offer presentation engine to provide personalized offers to shoppers via personal mobile devices. We will illustrate how the new services are designed to enhance in-store shopping experience with personalized real-time shopping information and ideas, and touch on how the experience can be continued across other channels.

This paper is organized as follows: Section 2 presents an offer presentation engine design. We will identify various shopper behaviors as system input; describe a set of keyword-based rules and the construction of the offer presentation engine. A novel layout scheme for organizing offers with respect to shopping cart contents is presented in Section 3. Finally, Section 4 concludes the paper with the scenarios of seamless multi-channel collaborative shopping.

2 In-Store Personalized Offer Presentation

In-Store offer personalization requires implicitly or explicitly collecting shopper information and leveraging that knowledge to decide what information to present to shoppers and how to present it. Since late 1990s, the world-wide web and the emergence of e-commerce have led to the development of personalization system and recommender systems. There are two main approaches that are used in delivery of personalized information: rule based approaches and collaboration filtering [3].

Rule-base personalization involved extensive domain knowledge setup, which can be difficult to manage, and is particularly so with a large product catalog [4,5]. *Collaborative filtering,* also sometimes known as *data mining,* has emerged as an approach to help remedy this. Collaboration filtering involves gathering user preference and behavior, and then uses that data to algorithmically produce personalized information for shoppers [6,7,8]. The disadvantage of collaborative filtering is that it needs a large body of data in order to produce reasonable personalization information.

The paper reports an *offer presentation scheme* with a keyword-driven rule engine, which analyzes and classifies shopper behavior in order to personalize the offer presentation. The key elements in the offer presentation engine are discussed in the following subsections.

2.1 Rule Engine Input Information

The input information is analyzed and classified by the engine to select offers. New offers are chosen, each time when there is a new input event. The following list summarizes several types of input information:

- Items and item quantities in the shopping cart
  The act of placing an item into the shopping cart triggers the computation of a new set of offers associated with every item in the shopping cart. Item quantity is also considered as an input.
- Price check and item removal
  Action of a shopper scanning an item to check for item price can be an indicator of interest in the item. Shopper removing an item from the cart can trigger offering of some alternative items.

- Shopper's in-store location
  Location specific offers are generated using shopper's in-store location with respect to the store layout.
- Shopper profile
  Typical examples which constitute a shopper profile are: 1) shopper preferences, which are supplied by the registered shopper from time to time, 2) shopper categorizations, which are given to the shopper based on the demographic information, such as age, income, education, occupation, marital status and home address, and 3) shopper's past shopping history.
- Shopper's current shopping list
  Shopper's current shopping list can be composed on the enterprise web site or in-store. The list can also be edited in-store when browsing through the weekly specials using the in-store wireless device.

2.2 Rules for Offer Presentation

To fit into an inexpensive PC-based server, our in-store rule engine employs a small number of rule types and utilizes only keywords to categorize shopper's preference and behavior. We will discuss these keywords and rules in the following subsections.

2.2.1 Keywords

Our rule engine uses keywords to describe and classify each shopper behavior in a shopping session. A keyword may activate rules to introduce more keywords. There are two main types of keywords: *global* keyword and *local* keyword, being used by the offer rules. *Global* keyword affects the offer rule throughout the entire shopping session. *Local* keyword only affects the current purchased item and has no impact on the offers for any previous purchased items. Both keywords are further classified in the following ways.

1. *Item keyword:* An item keyword is a global keyword that defines a primary event. It can be the item's UPC code or a set of keywords that best describe the particular item. For example: For an item with UPC Code = 02550080262, Item Description = *"Folgers Coffee, Classic Roast, Automatic Drip",* its global keywords can be "02550080262", "Coffee", and "Ground". For an item with UPC Code = "02550000034", Item Description="*Folgers Instant Coffee, Aroma Roasted",* its global keywords can be "02550000034", "Coffee" and "Instant".
2. *Category Keyword:* A category keyword is a local keyword, derived from a set of item keywords and is used to categorize multiple items into the same category. For example: For the two items in the previous example, their category keywords are both "Beverage".
3. *Relationship Keyword:* A relationship keyword is a local keyword, used to introduce other category keywords that have a predefined cross relationship with the original category. For example: "Beverage" and "Coffee" can result in a relationship keyword of "Dairy Cream".
4. *Scenario Keyword:* A scenario keyword is a global keyword, and is used to categorize one's shopping purpose. It is derived from a set of keywords and the count of each of the keyword. A count can be the number of appearances of a particular keyword. For example: multiple occurrences of "plastic" and "utensil" could introduce a scenario keyword of "party". Similarly, multiple instances of "hot dogs" and "pork chop" could result in a scenario keyword of "barbecue".

5. *Profile Keyword:* A profile keyword is a global keyword, and is used to identify a shopper. Profile keywords of a shopper can be self-described preferences, or be assigned according to the enterprise analytical results.

2.2.2 Rules

There are two types of rules used by the offer engine, *classification keyword rules* and *offer matching rules (personalization rules).* Figure. 2 illustrates the relationship between these rules. A *classification keyword rule* is used to expand the keyword set that is used to classify an item. While an *offer matching rule* is used to associate a set of keywords with an offer. The detailed definitions of these rules are given below.

1. ***Item keyword rules:*** This rule defines the global keywords for an item being scanned.

   Item UPC code → (keyword$_1$, keyword$_2$, ...)

2. ***Category keyword rules:*** This rule defines the category keywords based on a set of global and local keywords. The set of global keywords includes the newly added item's global keywords and previously added items' global keywords.

   keyword$_1$+keyword$_2$+... → (CategoryKeyword$_1$, CategoryKeyword$_2$, ...)

3. ***Relationship keyword rules:*** This rule defines the relationship keywords. Item quantity is also an element in this rule. Item quantity can be from zero to any positive value. When the quantity is zero, the rule is also known as replacement rule. A replacement rule is typically used to suggest an alternative item when shopper removed an item from cart.

   CategoryKeywords+ItemQuantity → (RelationshipKeyword$_1$, ...)

4. ***Scenario keyword rules:*** This rule defines the scenario keywords

   (keyword$_1$,quantity$_1$)+(keyword$_2$,quantity$_2$)+... → (ScenarioKeyword$_1$, ...)

5. ***Offer matching rules:*** Once no more new keyword can be introduced, the offer engine decides on what offers are selected for presentation using *offer matching rules.* A typical offer rule has the format of

   keyword$_1$+keyword$_2$+... → (Offer$_1$, Offer$_2$, ...)

   An additional *dynamic rule* can be added either by the enterprise or by the store manager to promote an over-stocked item. A dynamic rule has the following format:
   keywords+ItemInventory → (Offer$_1$, Offer$_2$, ...)

2.3 Offer Engine

In this section, we will discuss the design and implementation of the offer engine. We will first explain the offer rules processing flow to help readers understand the relationship between the keyword rules and offer matching rules.

2.3.1 Offer Rules Processing Flow

It is important to understand the flow of how these rules are executed and the purpose of the category keyword pool and the domain keyword pool. The detailed processing flow, as illustrated in Figure 2, is discussed in steps.

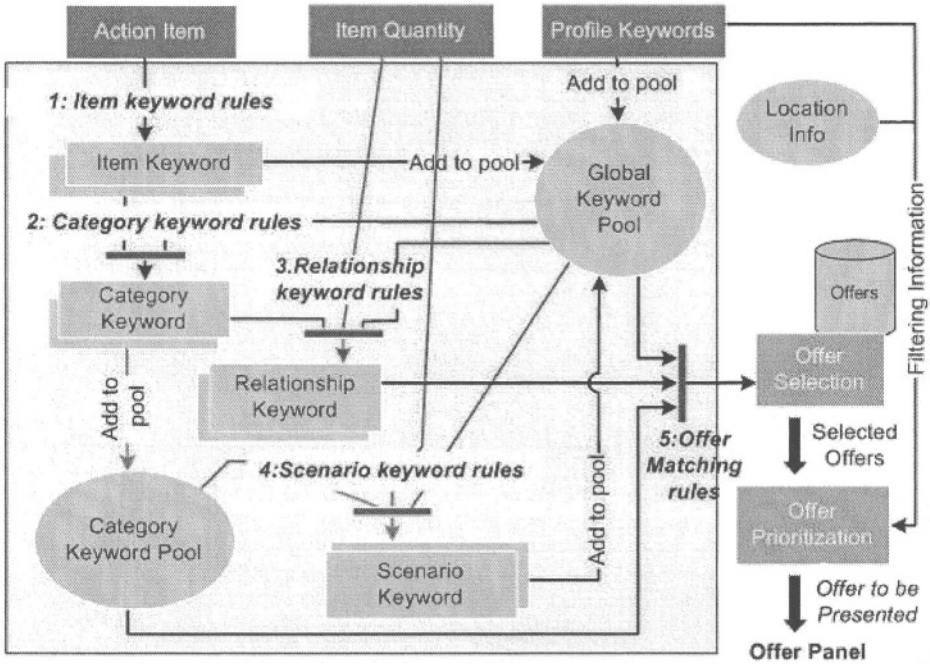


Fig. 2. Offer Engine and Offer Rules

Step 1 shows when an item is added into the shopping cart, a list of global keywords is derived using *item keyword rules.* Using this new list of global keywords and the keywords saved in the domain keyword pool, a list of category keywords is derived using *category keyword rules* as shown in Step 2. Depending on the item quantity, *relationship keyword rules* are applied to see if any relationship keyword can be added (Step 3). Combining item quantity, a list of category keywords and keywords from category keyword, *scenario keyword rules* are applied to see if any scenario keyword can be added.

For each of these steps, if a new keyword is introduced by a rule, earlier rules are re-executed to determine if any new keyword can be added. At this point, the rule engine will use the *offer matching rules* to see if any of the offer rules can be applied. There can be many offers eligible to be presented by the different combination of keywords.

Any newly acquired category keywords are added to the category keyword pool to be used later. The newly acquired item keyword and the scenario keyword are also added to the domain keyword pool to be used later as well. The domain keyword pool contains keywords that affect the entire shopping session, while the category keyword pool is used only to determine the scenario keyword.

2.3.2 Offer Engine Design and Implementation

Some of the key requirements of the offer engine are simplicity and efficiency. If an offer cannot be generated in real-time, the offer information might not be useful to a shopper, who may be walking away from the items to be promoted. To achieve this

goal, we developed an incremental rule scoring method to efficiently monitor if any offer rule can be fired.

Figure 3 shows the implementation of the proposed offer engine. Each item in the cart has an offer score list associated with it. Each score is initialized to the total number of keywords to be satisfied in an offer rule. Offer engine examines each keyword in the item classification keyword list and decrements the score of an offer that contains this keyword. After offer engine repeats the steps for each of the keyword in the item classification keyword list, it checks to see if any offer now has a score of zero. The offer with a score of zero means the lists of keyword that are needed for this offer, as specified by an offer rule, have all been found and this offer is selected.

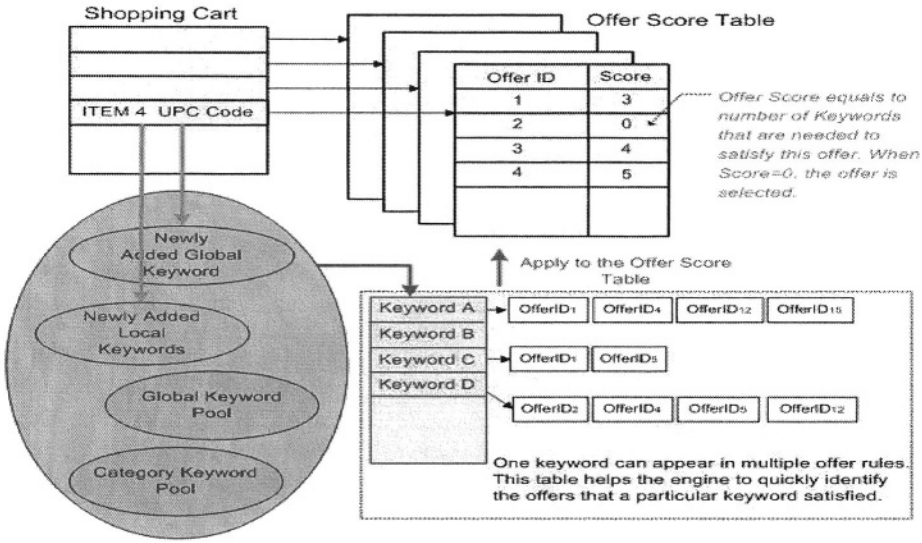


Fig. 3. Design and Implementation of the Offer Engine

Any newly added global keyword is also rechecked against other in-cart items' offer score lists. Any offer that has a score of zero after applying a global keyword is added as a new offer for that item. The objective is to ensure that a later added item's global keywords affect the offers of an earlier item in cart.

3 Offers Layout for Ease of Use

We have implemented a novel offer presentation method to display a set of promotion offers to shoppers in a personal shopping device environment. The method allows an efficient use of limited display area in a personal device. Item's offers are organized and displayed as a group for an item in the shopping cart. A highlight icon, such as a light bulb, is used to visually signal the shopper of the existence of a new (unread) collection of offers associated with an item. The collection of available offers is structured in such a way that allows a shopper to expand or collapse its views similar to Microsoft Windows file system tree-like structure using different icons (+ or – icons). Deleted items are first marked as delete, and then can be permanently removed by

activating a delete (x) icon. In Figure 4, we illustrate several scenarios of how a set of offers is displayed when using these icons.

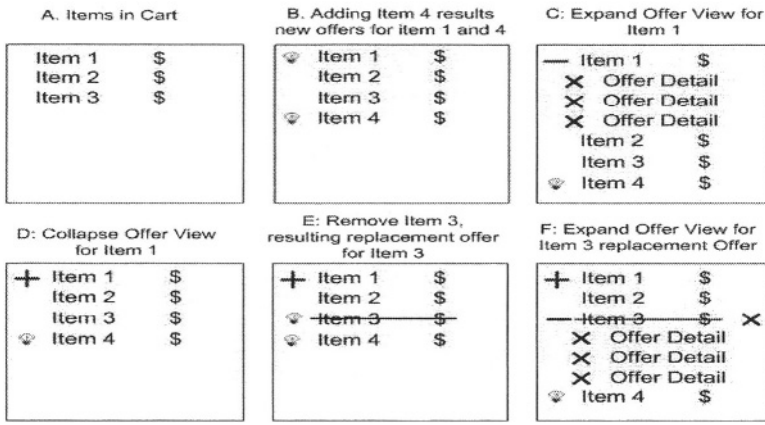


Fig. 4. Examples of Various Offer Presentation View

Due to the limited display space on a wireless shopping device, it may be undesirable to use the tree-like view to display offers that can occupy entire shopping cart area. Alternatively, multiple views of a separate offer panel may be used to allow shopper to switch views between a collection of offers associate with a selected item or with the shopper's location. Figure 5 shows a screen capture of such an implementation in a personal shopping device.

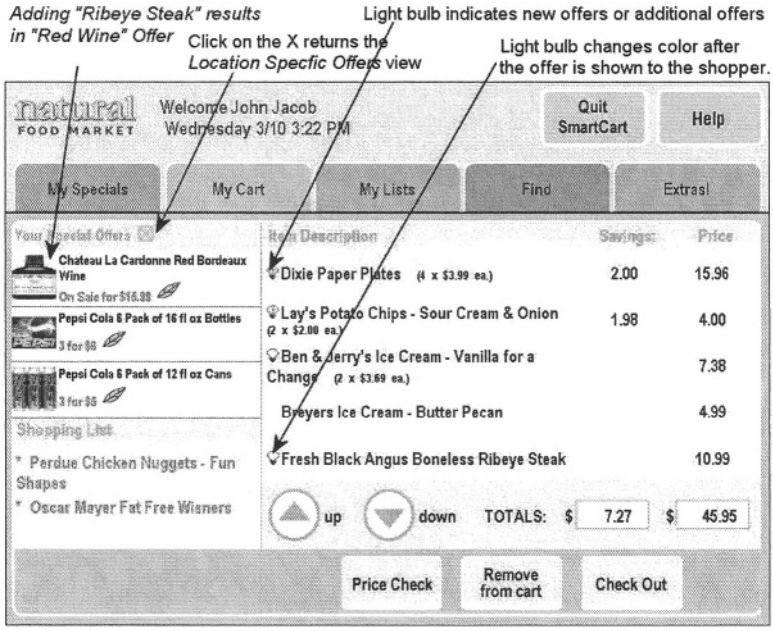


Fig. 5. Offer presentation implementation

4 Concluding Remarks

In this paper, we have presented an offer presentation engine for the in-store commerce server. It changes the way a retailer presents offers, from product push to consumer pull. By better understanding shopper's shopping preference, and tailoring to the specific shopping intentions, the selected offers suggest customers what they likely want, in the right timing and location.

Although the work presented here focuses on the in-store commerce server, it is relevant in the multi-channel environment. For example, offers presented in-store may not be limited to in-store items. An offer presented in-store can also be viewed and added into the future-shopping list which can be manipulated from the web channel. In addition to seamlessly ordering the product between web and in-store channels, it can also offer the collaborative shopping in the multi-channel environment. For example, a gift registry list build in the web channel can be presented to the shopper in store. Items purchased in the store can be immediately removed from the gift registry list. When shopping for a gift for a particular person, who is also a member of the store, the shopper can ask the offer engine to present a personalized gift idea list. Any member can choose to share a subset of his past purchase or his profile to the offer engine for offer presentation purpose only.

References

1. Smart Store: Enhancing the retail customer's shopping experience, IBM Executive Technology Report. IBM Business Consulting Services (2003)
2. Konstan, J. A. and Riedl, J. T.: Recommender systems in e-commerce, Proceeding of the 1st ACM conference on Electronic commerce, 1999 (158-166)
3. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce , Proceedings of the 2nd ACM Conference on Electronic Commerce (2000)
4. Anupam, V., Hull, R,. Kumar, B.: Personalizing E-commerce applications with on-line heuristic decision making, Proceedings of the tenth international conference on World Wide Web (2001)
5. Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: Effective personalization based on association rule discovery from web usage data, Proceeding of the third international workshop on Web information and data management (2001)
6. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms, Proceedings of the tenth international conference on World Wide Web, (2001)
7. Herlocker, J., Konstan, J. A., Terveen, L. G., Riedl, J. T.: Evaluating collaborative filtering recommender systems , ACM Transactions on Information Systems (TOIS), volume 22, Issue 1, (5-53)
8. Deshpande, M., Karypis, G.: Item-based Top-*N* recommendation algorithms, ACM Transactions on Information Systems (TOIS), Volume 22 Issue 1. (2004)

EXHIBIT E

RC24965 (W0910-096) October 16, 2009
Computer Science

IBM Research Report

Toward a Mobile Digital Wallet

Alan Cole, Scott McFaddin, Chandra Narayanaswami, Alpana Tiwari
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Toward a Mobile Digital Wallet

Alan Cole, Scott McFaddin, Chandra Narayanaswami, Alpana Tiwari
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
{colea, mcfaddin, chandras, alpana}@us.ibm.com

ABSTRACT

Mobile phones have now made their way into a large fraction of pockets and handbags worldwide. An intriguing question is whether such phones will eventually replace the physical wallets we carry. We believe the answer is in the affirmative, though plenty of challenges abound in overcoming entrenched personal and business practices and processes. In this paper, we explore the changes that need to ripple through the ecosystem to build a vibrant set of digital wallet services that potentially interact with each other to provide users both with increased convenience and a level of functionality hitherto unrealized. We describe our initial mobile wallet prototypes on web-enabled smart phones, designed to explore some of the challenges in creating the architecture and infrastructure necessary to make this vision a reality. Feedback from users and experts across a range of industries such as retail, banking, telecommunications, and healthcare indicate that we have just scratched the surface and a substantial wave of innovation is necessary to make the digital wallet a full-fledged reality.

Keywords

Mobile wallet, mobile applications, mobile computing systems, digital cash, coupons, receipts.

1. Introduction

As the mobile phone continues to take on an ever more central role in our lives, it is increasingly replacing old activities and practices. It is commonly accepted, for example, that the mobile phone is superseding the wrist watch [7], especially among the younger generation. Further, internet-enabled smart phones that also include cameras, music and video players, large storage, and navigation services are reducing the need to carry multiple devices. In response to these significant changes, we have been experimenting with ideas about how the mobile phone might also entirely replace the function of a physical wallet.

One of the first things to come to mind when a mobile digital wallet is mentioned is digital cash. [9] But when we examine our own wallets, we find many other objects there in addition to money. Among the principal functions of a wallet is to carry our identification such as a driver's license, employee badge, and other forms of id. We also carry credit and debit cards, loyalty cards, health insurance cards, and membership cards for the local library or professional association in our wallet. Wallet items are often grouped by function – for example, shopping lists, coupons, and receipts all support shopping. We might carry business cards in our wallet, both our own as well as those we receive from others. We may also use the wallet for purely personal functions – for example, carrying pictures of our children or pets, making the wallet into a miniature photo album.

Beyond the analogy with a physical wallet, a mobile digital wallet provides additional functions and benefits, such as virtually unlimited storage, location awareness, and quick sorting or searching of its contents, making it an even more compelling replacement for the physical wallet. Doing away with paper receipts, business cards and other paper artifacts, and the potential for optimizing or eliminating trips, all have environmental benefits. Note that some of the same security and privacy concerns associated with a physical wallet are also present for the mobile version -- losing either can be enormously disruptive.

We define a *mobile digital wallet* as a heterogeneous managed store of content items related to daily transactions, both electronic and physical, providing secure, automated multi-channel access to the user and other parties. What issues does the research community need to address to make a mobile digital wallet a reality?

This paper positions the emergence of a mobile digital wallet ecosystem as the central need, rather than just the need to build new capabilities into mobile devices. The paper first presents some advanced usage cases that are possible within such an ecosystem. We then analyze the players and roles involved in an ecosystem, and derive some design principles for it. We then outline technical capabilities that will be needed in its enabling software. We then describe a substantial customer pilot and mobile digital wallet prototype that backs our ideas. Additionally we summarize related work others have conducted in this area and outline our conclusions and future directions.

2. Digital Wallet Scenario

Jane is a busy working mother with three demanding young children. In a typical week, Jane visits several grocery stores, "Grocer-o-Rama" near her home, a more upscale "Fresh Foods" closer to her work place, and "Shop A Lot" near her children's day-care center but which usually has long lines on weekday evenings. Frequent trips to the local library, pediatrician's office, and a pharmacy further complicate her life.

But Jane has recently switched to a more advanced mobile phone that includes a futuristic digital wallet on board, and this has helped to simplify her life. The phone receives coupons and promotions, based on her profile and past shopping lists, which relieves her from needing to clip coupons from the Sunday paper and fumbling with them at checkout. She can see the estimated total cost of her current shopping list at the three different stores she patronizes, and pick the one that optimizes cost and travel time. The list itself is proposed, based on past lists and electronic receipts, and is easily modified to exclude or include specific items. Further, it is automatically sorted in aisle order when she enters a particular store, saving time and frustration as she shops. She pays for her purchase by selecting a credit or debit card from

her phone, which suggests a preferred card based on current balances, due dates, promotions, and interest rates. Sometimes she pleasantly finds that several of her expired coupons have been refreshed with valid coupons.

As she checks out by briefly touching her phone to the point of sale terminal while entering her PIN code, the electronic receipt and new promotional offers are automatically sent to her wallet, replacing the old-fashioned offers on a printed receipt, which she never seemed to be able to find again the next time she shopped.

Because Jane's new digital wallet makes use of a variety of services effectively located in the cloud, services that are able to interact with one another to provide intelligent suggestions and choices, she finds that her busy schedule is indeed simplified. Because she is now able to manage the family finances better and to eliminate unnecessary paper and optimize travel time and her own schedule, she feels that her new digital wallet represents a significant advance, an innovation with both ecological and financial benefits.

As we write this today, this scenario is still futuristic, with lots of unresolved questions and issues. However, we believe the vision is an appealing one; the next three sections outline some first steps toward achieving this vision.

3. Building a Mobile Wallet Ecosystem

We believe that a central hurdle for widespread acceptance of the mobile digital wallet concept will be the emergence of a generalized *mobile digital wallet ecosystem*. In this section, we lay out, as shown in Figure 1, what we consider some of the key players and needed capabilities in such an ecosystem.

3.1 Ecosystem Roles and Players

3.1.1 Wallet User. The most fundamental role is that of the wallet user. In this ecosystem, a digital wallet -- pictured at the center in Figure 1 –-- been established on behalf of a user. The digital wallet ecosystem fosters two key perceptions on the part of the user. First, the user has a perception of ownership and control of the digital wallet, even if the maintenance and technical control of the digital wallet is in the hands of another party. Second (assuming that the user generally makes use of a mobile device), the user has a perception that the wallet exists inside that device. The role of the user is essentially to manage the wallet as an item of property similar to an email account. Even though the wallet is conceptually attached to the device, the user should be able to access the wallet through multiple channels (e.g. mobile web, desktop) and use the wallet within transactions conducted both online and in physical settings such as stores.

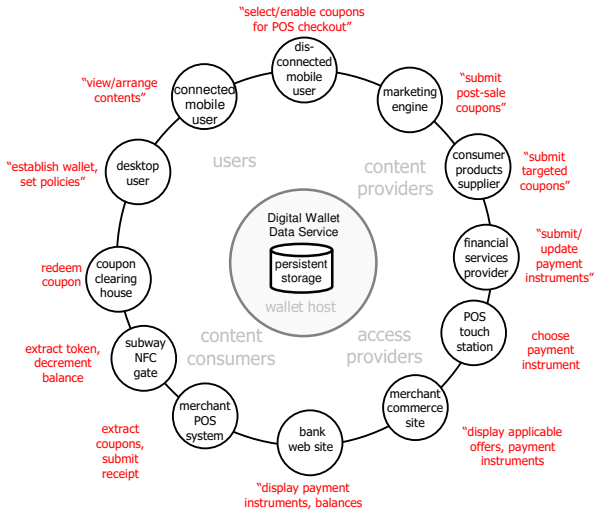


Figure 1: Illustrating elements of a mobile digital wallet ecosystem.

3.1.2 Wallet Content Providers. Entities playing this role wish to provide objects for inclusion in the wallet. For example, a bank or other financial services provider may wish to provide a debit or credit card as a content item for the given user's digital wallet. An online marketing company may wish to provide digital coupons or offers. An advertising company may wish to provide targeted advertisements to the digital wallets of select users. In most cases, wallet content providers operate under the control of the wallet user. In many cases, the user will desire the inclusion of the content provider's wallet items (e.g. the user's bank account should be made available within the wallet) and in many other cases, the user will view the content provider as a spammer. As a result, the wallet should offer a spectrum of access control capabilities that easily incorporate the content items of value to the user, reject those of no value, and provide some intermediate capability for items of possible value.

From the content provider's point of view, there are several key hurdles to overcome. First, there must be a way of identifying the wallet for a particular user. Second, there must be a way of acquiring the necessary credentials for interaction with the wallet of the user. Third, there must be a way of introspection over the wallet so that the rules for accessing the wallet can be determined in advance of dedicating resources to working with the wallet.

3.1.3 Wallet Content Consumers. These are entities that wish to read, update, or extract an item from a wallet. For example, a point of sale (POS) station deployed by a merchant may wish to read the set of payment instruments available in a user's wallet (after authentication of itself to the wallet) and display the available options to the user (e.g. on a touch screen). In some cases the content consumer will disable the wallet item (e.g. a "use once" coupon), mark it (e.g. a multi-use coupon), or decrement it (e.g. a digital cash balance). In other cases the content consumer may be enabled to remove the content item from the wallet altogether.

3.1.4 Wallet Host. In general, someone must implement the wallet and make it available to the other players in the ecosystem. There has been much debate and tension in the industry over this role.

For example, from a mobile usage point of view it would appear to be a natural role for a mobile telco. From the vantage point of classical electronic commerce, it may appear that this role would belong to retailers, banks or financial services providers.

3.1.5 Wallet Access Providers. Beyond the question of who hosts the wallet, arises the question of who provides access to it. One natural role is for the host to the *primary* access provider. For example, the host would provide desktop and web access screens that a user can use to manage their wallet contents, as well as services oriented interfaces. However, within the ecosystem many other players may wish to assume the role of *secondary* access providers within the context of their business function. For example, a merchant may wish to introspect over a user's wallet, read coupons or offers that apply to the merchants product set, and display those applicable items within the shopping or checkout screens of that merchant's online store. As the ecosystem evolves, the roles of various access providers may blend together, and possibly will disappear altogether, with the digital wallet disappearing into the background of everyday commerce.

While we have outlined several distinct roles, in practice, many players in the ecosystem will likely play hybrid roles. A merchant, for example, may be a content provider, content consumer, and access provider.

3.1.6 Standards Organizations. A critical consideration in enabling services from different providers to interoperate is the establishment of industry standards. While some efforts in electronic commerce have been fruitful, there are still many gaps; for example, there is still no accepted standard for the content of a digital coupon.

3.2 Design Considerations

From the analysis of the above roles, we can extract some key design considerations for the digital wallet.

3.2.1 Wallet as a centralized managed service. The wallet should be positioned centrally in any ecosystem, allowing access to a wide field of players. Towards this end, we note that there is increasing attention to the concept of cloud-based services, in which core computational services and business processes are migrated into centrally managed common assets. The wallet ecosystem is ideal for such a service structuring. In particular, a digital wallet is likely not to be an item of licensed software, but a managed service. The players in the ecosystem would become subscribers to a cloud-based wallet service.

3.2.2 Heterogeneity of the digital wallet. The wallet ecosystem should accommodate an arbitrary collection of content types, following a design principle similar to that of the web. Basic access capabilities should be managed in a uniform way that is independent of content type. The ecosystem should provide extensibility mechanisms that accommodate type-specific constructs such as business logic and user interactions. Ecosystem players should agree upon a model for interchange of wallet items and forms of access control.

3.2.3 Wallet as competitive asset. A digital wallet should be managed as a competitive asset. This means that a wide variety of players will need to (1) insert items into particular locations within the wallet, (2) have those items securely managed by the user or by agents that operate on behalf of the user, (3) update (or mark) items in the wallet, and (4) extract or delete wallet items. Digital wallets will need to be managed much like email accounts, and will be vulnerable to an unlimited number of demands for access and equally susceptible to the same motives with regard to spamming.

3.2.4 Wallet Identification. Since the wallet ecosystem virtualizes the wallet, it does not actually reside on the user's physical device. Thus, we cannot identify digital wallets via MAC address, SIM card, or MTN. Another approach may be to identify digital wallets via an email addresses (e.g. user@wallethost.com). However, this has the disadvantage of linking the wallet to a particular provider, and introduces the same problem with email – users typically have multiple accounts. Another approach might be to use a directory mechanism -- however, these approaches tend to be cumbersome and likely beyond the tolerances of end users.

3.2.5 Wallet Lifespan. As the digital wallet incrementally replaces the physical wallet, questions of its lifespan are induced. For example, how long does a digital wallet live? Who owns the contents of a digital wallet? If the user dies, who recovers ownership of the digital wallet contents? Service changes such as switching wireless service providers should be seamless and not dramatically affect the usage of the wallet.

3.2.6 Wallet Automation. Common usage patterns should be managed in a silent, unobtrusive, and automatic fashion. These usage patterns should be driven by a "*hands off*" design point: it should be possible to use your digital wallet in your everyday mobile commerce life with a minimal use of your hands, except for trivial steps such as swiping it at an NFC service point. A digital wallet should allow the attachment of user-preferred agents that execute automated processes in well recognized mobile commerce settings. For example, an agent for the retail POS check out process would automatically carry out scripts which identify the merchant, collect applicable instruments from the wallet (coupons, payment instruments, loyalty cards) and organize a silent, single step transaction with the merchant.

3.2.7 Wallets in Workflows. The emergence of a digital wallet ecosystem may lead to new models of processing transactions. For example, in the payments world a typical payment is viewed as the presentation of a payment instrument by the user to a receiver (e.g. retailer), followed by an opaque process of routing payment transactions through a series of financial providers. With a digital wallet, each of these providers could be modeled by an individual wallet (e.g. your bank has a wallet) and this process may be recast as the successive transfer of content from one wallet to another.

3.2.9 Interacting Wallet Services. Mature mobile wallets will become increasingly heterogeneous, managing diverse individual artifacts and corresponding services. Users will realize multiplicative rather than additive benefits when these services interact with each other. This principle is already emerging in existing mobile device usage – for example, composable functions which link the user's address book to the navigation application on the device allow users to get driving directions by clicking on a contact in the address book. A map in a navigation application in turn can be used to enter an address for a contact. Coupons, promotions, advertisements, receipts, loyalty program, and payment services within wallets must interact with each other and with native device services even when they are provided by multiple service providers.

3.2.10 Monetization of the Ecosystem. Central to the adoption of such an ecosystem is the resolution of issues regarding who pays for and benefits from its functions. An inevitable axiom is that users should perceive value and benefit. Beyond that, a wide variety of monetization schemes could be envisioned. For example, valuations could be attached to each basic operation against an item in a digital wallet. Users may be willing to pay for items as they are added to a wallet (e.g. pay 5 cents for a coupon offering a 50 cent discount), or as they are extracted (e.g. when the user redeems a previously stored coupon). Marketing firms may also receive benefit when the user accepts an item into a wallet – the equivalent of receiving payment for each page view of an advertisement. Wallets could also be monetized based on user access alone in the way that many email providers make money -- either by associating advertising with account access, or through paid subscriptions for advanced accounts. Novel hybrid functions that add value to all parties could be monetized – for example, by attaching coupons to certain payment instruments in a digital wallet, a user may be influenced to use one credit card vs. another.

3.2.11 Client Programming Model. One of the ongoing debates in the mobile application area is whether browser-based web applications or native applications are better [5]. A browser-based digital wallet will work across a wider variety of devices, but is handicapped by different browser implementations and limited access to phone capabilities, such as a camera that can serve as an input device or GPS that can provide user context. Native digital wallet applications are generally better performing and have access to all the device capabilities, but developers face the problem that creating a version of the application for each of the many devices is a huge amount of work, and requires users to download and update applications. Though the advent of HTML 5 [6] within mobile browsers will give an impetus to the web-based approach, we advocate a hybrid approach in which much of the presentation and user interaction is done with browser-based web pages, but in which bridges to native code are defined to give access to device capabilities.

4. Design for Mobile Wallet Service

The preceding section postulated a centralized managed service at the heart of the digital wallet ecosystem. This section outlines the technical capabilities that we view as critical for its success.

4.1.1 Basic User Management Capabilities. As noted above the digital wallet should be a user managed store for a heterogeneous collection of content items, centrally managed and made visible throughout an electronic commerce ecosystem. The wallet should be accessible by users through a variety of access channels, including the desktop web, mobile web, and native mobile applications. The wallet should provide basic user driven operations for searching/sorting and finding wallet items, plus additional operations for user controlled item insertion, modification, and deletion. Key user capabilities should include management of intelligent views across the wallet (the wallet may be a large data space). Views of the wallet should accommodate pluggable presentation modules, which present both previews and detailed views of wallet data types in customized ways. Adding a new type of content item to a wallet should entail no more than plugging in additional display and management modules for that content type.

The wallet should provide versatility in its basic paradigm of organization. We recommend that the wallet offer a naming and organizational model that provides the capabilities of both a directory/folder mechanism and a tag/attribute mechanism. This allows the user to establish the concept of "a place" that a wallet item exists in the wallet (e.g. "my Coupon folder") plus the concept of "properties" within the wallet (e.g. "my expired coupons"). Possible enhancements for mobile commerce could include the ability to manage wallet items by geo-coded locations.

In many cases the wallet is used in settings which have space limitations (e.g. displayed on a small device screen) and or constraints about the complexity and timing of user operations. For example, the user needs to quickly access the digital wallet while standing in a checkout line at a point of sale station. Therefore, the wallet should offer organizational capabilities such as filtering and prioritization of wallet items. To address a wide variety of unpredictable needs, these organizational capabilities should be structured as pluggable modules. For example, a merchant may wish to offer a pluggable wallet organizer which automatically sorts the user's coupons by aisle when the user enters that merchant's store.

4.1.2 Access Control Capabilities. Access to the wallet should be governed by a set of access rules which determine who can carry out operations (find/add/delete/update) on wallet items. These rules should be customized by data type. A simple conceptual model for such an access control rule would be a tuple: *(operation, location, content type name, content type namespace, credential type, credential)*. The operation member would be one of the values `{find,add,delete,update}`, the location member would be a folder name, the content type and namespace member would identify which type of item can be manipulated by the operation, and the credential type and value members would specify which actors can carry out the operation. For example, the rule `("Add", "/Coupons/Inbox", "Coupon", "http://coupons.org", "certificate", …)` would specify that only callers which present a given certificate can add coupons to the wallet.

4.1.3 Work processes. In addition to access control capabilities, which determine what content gets into the wallet in the first place, additional work processes should be in place to manage wallet contents. For example, the wallet should have basic management capabilities that automatically recognize and remove expired artifacts (old coupons, unused offers, etc.). Another work process should also be in place to limit the maximum size of the wallet contents.

4.1.4 Event and Notifications. The wallet implementation should be capable of generating and managing events related to wallet operations. In a simplified form, a wallet can be viewed as the source component of an event stream tied to the four basic operations (find/add/delete/update). This allows logical software components to be attached to event streams flowing from a wallet, for example, summary notifications may be sent to the wallet user whenever certain artifacts are added (e.g. "you have three new coupons from Manufacturer X").

4.1.5 Replication and Content Transfer. As noted in the ecosystem model, a wallet would be managed in a central location and accessed through various channels and devices. In some cases local copies of a wallet view would need to be maintained outside of the central wallet. For example, native wallet applications on

mobile devices could operate against replicated snapshots of the user's wallet, enabling usage in situations where network connectivity to the central wallet service is unavailable or unreliable. The wallet service should support replication strategies where localized copies are synchronized at opportune times. Entities that accept replicated artifacts from mobile wallets should have the opportunity to verify the artifact before finalizing its acceptance. For example, a coupon presented from a replicated wallet would essentially be *a claim* about the true artifact stored at the central wallet. In such a scenario, the user's wallet may be disconnected but the receiving merchant may be connected and can verify the coupon. It should also be possible to transfer items from one wallet to another. Transfer capabilities should be in effect even if the two wallets are not managed by the same wallet host.

4.1.6 Sharing. In many usages, the contents of a wallet could be shared among users – for example, family members may wish to share payment instruments or virtual cash. This can be realized in two ways: replication of the shared items from a base wallet, with periodic synchronization, or via virtualized views on a base wallet. Operations carried out on a virtualized wallet view would be transferred to the base wallet. Virtualization storage schemes would induce the need for virtualized access control policies, for example, a conservative scheme that computes and enforces the maximum access requirements among the base and virtualized wallets.

4.1.7 Staged import and export. The combination of the above wallet storage, access control, and replication and transfer capabilities can be combined in useful ways that yield higher level policies about wallet usage. For example, access rules could be arranged to allow artifacts to be imported only into special staging folders (e.g. "`/Coupons/Inbox` These can subsequently be accepted into production folders (e.g., "`/Coupons/Groceries`") either manually or by an automated agent. The same staging approach could be used for exports, with relevant artifacts moved into special export folders depending upon the user's activities. This could be especially useful for speeding wallet usage at critical times such as point of sale. For example, a retail workflow agent could recognize which store the user is shopping in, establish a transient folder (e.g. "`Exports/XYZStore.Nov212010`") for staging exports, then move applicable coupons into the "`Coupons`" subfolder, applicable payment instruments into the "`Payments`" subfolder and the retailers loyalty card into the "`LoyaltyCards`" subfolder. The user could then quickly review the items and approve the items before approaching the point of sale station, allowing the transaction to be completed in a single step. The staging folder could be referenced via a generated short code (e.g. "`ABC123`") and presented to the POS operator or, in the presence of NFC, a ticket datum referencing the export folder could be exchanged with the POS.

4.1.8 Self description and semantic linkage. A heterogeneous wallet requires that content items within be self-describing, at both the schematic level (fields and values) and semantic level (meaning). This capability is particularly important in enabling the hybridized business functions envisioned in the ecosystem such as an agent suggesting shopping strategies based on receipts and coupons in the wallet) and also in enabling ad-hoc operating modes. For example, a self describing wallet can be presented to a merchant with no prior administration. Such cooperating services are at the core of the semantic web and will likely gain equal importance in a mobile digital wallet ecosystem.

5. Prototype and Customer Pilot

As a test of the concepts described above, we developed a substantial prototype implementation of a mobile digital wallet and recently conducted a customer pilot with a major U.S. retailer. The customer pilot involved a digital wallet scenario that combined the function of two wallet artifacts: (1) loyalty cards and (2) store promotions (e.g. coupons and offers). Using this solution, loyalty program customers may use their mobile devices when they visit a pilot store to receive price promotions.

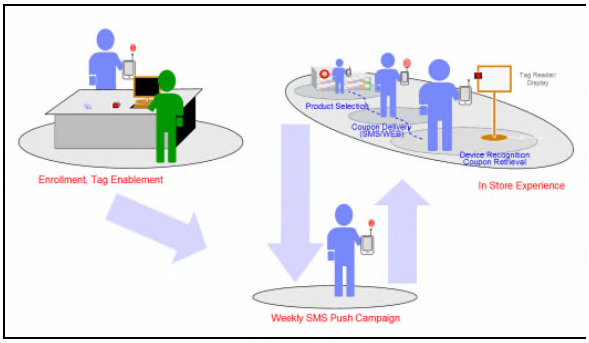


Figure 2: Illustrating three phases of the customer workflow for in-store mobile digital wallet pilot.

The pilot involved three phases of usage, seen clockwise in Figure 2. The first phase is an enrollment phase in which a mobile digital wallet is established for a user and linked to their existing loyalty card. A wafer-sized RFID tag, used for customer recognition, is attached to their mobile phone. The mobile digital wallet is stored on a server, consistent with the "wallet-as-service" principle outlined above. The second phase is the in-store visit: the customer enters the pilot store, swipes the mobile device at an RFID reader station, receives in-store offers into the digital wallet, and is notified of new offers via SMS messages. Additionally the user can access their digital wallet from their device using a mobile web application. Discounts are automatically applied at the point of sale when the user presents their loyalty card. The device is not involved at the point of sale. The third phase is an opt-in reminder campaign in which weekly text messages remind users of the week's promotions.

Figure 3: Prototype of general purpose wallet running on Android phone.

The pilot was motivated by the retailer's desire to find new ways to strengthen their relationship with loyalty cardholders. We tracked user participation and offer redemption rates during the pilot. We found coupling business processes such as marketing campaigns with the on-device wallet led to significant business

advantage. Though confidentiality agreements prevent us from releasing detailed results, we found that (1) the frequency of in-store visits was greater than the visit rate of the baseline loyalty program, and (2) the electronic coupon redemption rate was several times higher than traditional paper coupon redemption rates.

The success of the pilot motivated us to broaden the digital wallet concept and forms the basis for many of the assertions described in this paper. Specifically, we generalized the prototype to support an arbitrary set of content types, rather than the special purpose ones used in the pilot. Figure 3 illustrates this prototype configured for mobile coupons and electronic receipts. At present our prototype supports pluggable organizational modules which arrange wallet artifacts by a variety of parameters. In addition to the ability to arrange the wallet by simple sorting and searching on fields such as amount or expiration date, pluggable modules provide the ability to arrange artifacts based on complex concepts (e.g. order coupons by store aisle or display receipts by geocoded locations). Pluggable display elements induce custom visuals for content types. Our current client model is browser-based, however we are working towards integration of the wallet client with native capabilites on the device such as camera (facilitating in-store barcode scanning, which can be used to correlate wallet items to products) and GPS (facilitating the correlation of wallet items to location). In each of these versions, we have used a centralized managed service model for the wallet (following many of the capabilities described in Section 4), built on our testbed system called Celadon [10].

6. Related Work

Much of the previous work in this area has been concerned with mobile payments—the use of the mobile phone as a surrogate for a credit card or smart card. The device is linked to a bank account or credit card or phone account; alternatively, it may be supplied with a fixed amount of digital cash, which may then be spent anonymously [4]. In some regions, this use of mobile phones for payment is already well established. In Japan, Hong Kong, and Singapore, for example, the FeliCa contactless RFID chip is included in many mobile phones, enabling mobile payments. In South Korea, mobile payments are well established; contactless RFID chips or SIM-sized cards inserted into the phone enable mobile payments, with charges showing up on the customer's phone bill [3]. Taking a somewhat different approach, PayPal Mobile allows certain types of transactions from a mobile phone. However, mobile payments alone do not constitute a mobile wallet, as evidenced by the personal observation that consumers in South Korea and Japan still carry the old-fashioned kind of wallets. A somewhat broader approach is demonstrated by prototypes that combine mobile payments with organization of credit and loyalty cards, based on an NFC phone [1][2]. In other efforts, there are now numerous commercial or free mobile applications to store and organize various individual content types, for example passwords, loyalty cards, shopping lists, business cards, coupons, and so on [8]. But a series of applications, each dealing with one or two different content types, each with a different user interface, each possibly requiring a separate login, falls far short of what we believe is required to make the mobile phone a viable replacement for the physical wallet. We believe that to accomplish this goal requires a unified architecture, able to accommodate an open set of content types. Standards will also be an important aspect of this work, enabling independently-developed services from multiple providers to interoperate with one another.

7. Conclusions and Future Work

We have described the opportunities and challenges that arise in designing a mobile wallet that could potentially replace a physical wallet. Challenges abound in user interfaces, business models, standards, interactions among wallet contents and services, exploiting user context, identifying user intent, lifecycle management, etc. Our initial implementation of a digital wallet holds coupons, loyalty cards and electronic receipts. In the future, we plan to expand our repertoire of services to include a larger catalog of content types, refine the user interface, and expand it to a wider variety of user pilots.

8. References

[1] Balan, R. K. and Ramasubbu, N. 2009. The Digital Wallet: Opportunities and Prototypes, IEEE Computer, Vol. 42, No. 4 (Apr. 2009), 100-102.

[2] Balan, R. K., Ramasubbu, N., Prakobphol, K., Christin, N. And Hong, J. mFerio: The Design and Evaluation of a Peer-to-Peer Mobile Payment System, Proceedings of the 7th Annual Conference on Mobile Systems (MobiSys '09), June 2009, Kraków, Poland, pp. 291-304.

[3] Bradford, T. and Hayashi F. 2007. Complex Landscapes: Mobile Payments in Japan, South Korea, and the United States, payments system research briefing, The Federal Reserve Bank of Kansas City, http://www.kc.frb.org/PUBLICAT/PSR/Briefings/PSR-BriefingSept07.pdf

[4] Chaum, D. 1991. Numbers Can Be a Better Form of Cash than Paper, Computer Security and Industrial Cryptography 1991: 174-178.

[5] Fling, B. 2009. Mobile Design and Development, 1st Edition. O'Reilly Media, Inc.

[6] Hickson, I. et al. 2009. HTML5: A Vocabulary and associated APIs for HTML and XHML, W3C WHATWG draft, http://dev.w3.org/html5/spec/Overview.html.

[7] Irvine, M, Watches lose ground to cell phones, MSNBC Technology & Science, Feb. 16, 2007. http://www.msnbc.msn.com/id/17189064/

[8] Landay, J., Joseph, A., and Reynolds F. 2009. Smarter Phones, IEEE Pervasive Computing, 8, 2 (Apr.-Jun. 2009, 12-13.

[9] Mallat, N., Matti R., and Tuunainen, V.K. 2004. Mobile Banking Services, COMMUN ACM, 47, 5 (May 2004), 42-46.

[10] McFaddin, S., et al. 2008. Modeling and Managing Mobile Commerce Spaces using RESTful Data Services. IEEE Intl. Conf. on Mobile Data Management, 2008.

EXHIBIT F

Microsoft

OVER 10,000 ENTRIES

Microsoft Computer Dictionary

Fifth Edition

- *Fully updated with the latest technologies, terms, and acronyms*
- *Easy to read, expertly illustrated*
- *Definitive coverage of hardware, software, the Internet, and more!*

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Computer Dictionary.--5th ed.
p. cm.
ISBN 0-7356-1495-4
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002
004'.03--dc21 200219714

Printed and bound in the United States of America.

2 3 4 5 6 7 8 9 QWT 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com*.

Acquisitions Editor: Alex Blanton
Project Editor: Sandra Haynes

Body Part No. X08-41929

high-end scanners and allows retrieving of still images from IEEE 1394-based DV camcorders and USB-based Web cams. *Acronym:* WIA.

Windows IP Configuration *n.* *See* Winipcfg.

Windows Management Instrumentation *n.* A management infrastructure in Windows that supports monitoring and controlling system resources through a common set of interfaces and provides a logically organized, consistent model of Windows operation, configuration, and status. *Acronym:* WMI. *See also* resource.

Windows Me *n.* Released in 2000, the Windows Millennium Edition (Windows Me) operating system designed for home users as an upgrade from Windows 95 or Windows 98. Windows Me offers an improved home user experience including making it easier for users to share and manipulate digital photos, music, and videos, enhanced home networking capabilities, a rich Internet experience with support for broadband connections, different Internet communication tools, and online gaming.

Windows Media Audio *n.* A digital audio coding scheme developed by Microsoft that is used in distributing recorded music, usually over the Internet. Windows Media Audio shrinks the size of the audio file by a factor of 20 to 24 without seriously degrading the quality (CD-recording level) of the sound. Windows Media Audio files are given the file extension .wma and can be created with Windows Media Tools and played with the Windows Media Player. *Acronym:* WMA. *See also* Windows Media Technologies. *Compare* MP3, RealAudio, Secure Digital Music Initiative.

Windows Media Encoder *n.* A Windows Media technology that compresses live or prerecorded audio and video into a Windows Media stream, which can either be distributed immediately or saved as a Windows Media file for later distribution. The technology allows content developers to convert both live and prerecorded audio, video, and computer screen images to Windows Media Format for live and on-demand delivery. Windows Media Encoder also can save a stream as a Windows Media file and convert a file into Windows Media Format. Windows Media Encoder can distribute a stream via HTTP protocol. *Also called:* (if context is clear) Encoder, the encoder, the encoder engine.

Windows Media Player *n.* A client/control that receives a stream from a Windows Media server or local content for playback. It can run as a stand-alone client executable program. Windows Media Player can also be embedded in a Web page, a C++ program, or a Microsoft Visual Basic program that uses the client ActiveX control.

Windows Media server *n.* A server on which Windows Media Services has been installed.

Windows Media Services *n.* A digital media platform that runs on a server, such as Windows 2000, to support streaming media, such as video and audio.

Windows Media Technologies *n.* Microsoft technologies for the creation, delivery, and playing of streaming audio and video over a network, including both intranets and the Internet. Windows Media Technologies, downloadable from the Microsoft Web site, support both live and on-demand (delivered from storage) content and are based on files delivered in Advanced Streaming Format (ASF). Three major components—Windows Media Tools, Windows Media Services, and Windows Media Player—comprise Windows Media Technologies. See the table. *See also* Advanced Streaming Format. *Compare* RealSystem G2.

Table W.2 ATA Specifications.

Component	*Purpose*	*Features*
Windows Media Tools	Content creation	ASF authoring and editing tools, including tools for converting files from other formats (WAV, AVI, MPEG, and MP3) to ASF.
Windows Media Services	Content delivery	Tools for real-time and on-demand content delivery, administration tools, and Windows Media Rights Manager for piracy control.
Windows Media Player for PC platforms, Windows Media Player for Macintosh, Windows Media Player for UNIX	Content playback	ASF player for audio, audio plus still images, and full-motion video. Also supports other multimedia data, including RealAudio.

on an open architecture to allow new server components to be installed in any part of the stream at any time.

Wireless Transaction Protocol *n.* A lightweight request/reply transaction protocol for devices with limited resources over networks with low to medium bandwidth. It is not called the Wireless Transport Protocol or the Wireless Transfer Protocol. *Acronym:* WTP.

Wireless Transport Layer Security *n. See* WTLS.

wire-pin printer *n. See* dot-matrix printer.

wire-wrapped circuits *n.* Circuits constructed on perforated boards using wire instead of the metal traces found on printed circuit boards. The stripped ends of insulated wires are wrapped around the long pins of special wire-wrapped integrated circuit sockets. Wire-wrapped circuits are generally handmade, one-of-a-kind devices used for prototyping and research in electrical engineering. *Compare* printed circuit board.

wiring closet *n.* A room or location in a building where telecommunications and/or networking equipment such as hubs, switches, and routers are installed. *Also called:* data closet, telecom closet, telecommunications closet.

wizard *n.* **1.** Someone who is adept at making computers perform their "magic." A wizard is an outstanding and creative programmer or a power user. *Compare* guru, UNIX wizard. **2.** A participant in a multiuser dungeon (MUD) who has permission to control the domain, even to delete other players' characters. *See also* MUD. **3.** An interactive help utility within an application that guides the user through each step of a particular task, such as starting up a word processing document in the correct format for a business letter.

wizzywig *n. See* WYSIWYG.

WLAN *n. See* wireless LAN.

WMA *n.* Acronym for **W**indows **M**edia **A**udio. *See* Windows Media Audio.

.wmf *n.* A file extension that identifies a vector image encoded as a Microsoft Windows Metafile.

WMF *n.* **1.** *See* Windows Metafile Format. **2.** Acronym for **W**ireless **M**ultimedia **F**orum. A consortium of technology companies formed to promote open standards for wireless streaming products. WMF members include Cisco Systems, Intel, and the Walt Disney Internet Group. *See also* ISMA.

WMI *n. See* Windows Management Instrumentation.

WML *n.* Acronym for **W**ireless **M**arkup **L**anguage. A markup language developed for Web sites that are accessed with microbrowsers on Wireless Application Protocol (WAP)–enabled devices. A Web site written with WML would be viewable on handheld devices with small screens, such as cell phones. *See also* markup language, microbrowser, Wireless Application Protocol.

WMLScript *n.* A scripting language derived from the JavaScript language for use in the development of Wireless Markup Language (WML).

WMM *n. See* Windows Movie Maker.

word *n.* The native unit of storage on a particular machine. A word is the largest amount of data that can be handled by the microprocessor in one operation and also, as a rule, is the width of the main data bus. Word sizes of 16 bits and 32 bits are the most common. *Compare* byte, octet.

Word *n.* Microsoft's word processing software, available for the Windows and Macintosh platforms. In addition to extensive editing, formatting, and customization features, Word provides such tools as automatic text completion and correction. The most recent version, Word 2002 (part of Office XP) adds Web functionality—for example, the ability to save documents in HTML format. The first version, Microsoft Word for MS-DOS 1.00, was introduced in 1983.

word-addressable processor *n.* A processor that cannot access an individual byte of memory but can access a larger unit. In order to perform operations on an individual byte, the processor must read and write memory in the larger unit. *See also* central processing unit.

WordPerfect Office *n.* A suite of business application programs from Corel Corporation. The basic (Standard Edition) WordPerfect Office suite includes the WordPerfect word processor, Quattro Pro spreadsheet, Corel Presentations presentation software, CorelCENTRAL personal information manager, Microsoft Visual Basic for Applications scripting tools, and Trellix Web publisher. A home and small-business package, the Voice-Powered Edition, adds speech recognition and publishing products; a business and corporate package, the Professional Edition, adds database and Internet tools to all of the preceding.

word processing *n.* The act of entering and editing text with a word processor. *Acronym:* WP.

word processor *n.* An application program for creating and manipulating text-based documents. A word processor is the electronic equivalent of paper, pen, typewriter, eraser, and, most likely, dictionary and thesaurus. Depending on

EXHIBIT G

NEWTON's TELECOM DICTIONARY

24th Edition

Harry Newton

New York

NEWTON's TELECOM DICTIONARY

email: Harry@HarryNewton.com
book site: www.NewtonsTelecomDictionary.net
personal web site: www.HarryNewton.com

Published in the United States by
Flatiron Publishing
50 Central Park West
New York, NY 10023

email: Harry@HarryNewton.com
www.FlatironBooks.com
Printed by Port City Press, Cadmus – a Cenveo Company
1323 Greenwood Road
Baltimore, MD 21208

Distributed in the United States by
National Bank Network
4501 Forbes Boulevard, Suite 200 Lanham, MD 20706

Orders Phone toll-free 1-800-462-6420
Direct 1-717-794-3800
Fax 1-800.338.4550
custserv@nbnbooks.com

Distributed outside the United States by
Elsevier Inc.
11830 Westline Industrial Drive
St. Louis, MO 63146 USA
email: i.internet@elsevier.com
Tel: 1-314-453-7010
Fax: 1-314-453-7095
Toll-free 800-460-3110

ISBN 13 digit Number: 978-097938-731-9
March 2008 Twenty Fourth Edition
Steve Schoen, Contributing Editor
Gail Saari, Layout and Production Artist
Gail Saari, Saul Roldan, and Damien Casteneda, Cover design

Newton's Telecom Dictionary 24th Edition March 2008

Stay In Touch

For suggestions, corrections, updates, special offers: please send an email to Harry@HarryNewton.com

I promise you I won't give your name to anybody. Nobody. Promise.

Harry Newton

(CCSA) network. See Advanced Private Line Termination.

APM 1. Average Positions Manned, the average number of ACD positions manned during the reporting period for a particular group.

2. Advanced Power Management. A specification originally sponsored by Intel and Microsoft to extend the life of batteries in battery-powered computers. The idea of the specification is for the application programs, the system BIOS and the hardware to work together to reduce power consumption. An APM-compliant BIOS provides built-in power management services to the operating system of your PC. The operating system passes calls and information between the BIOS and the application programs. It also arbitrates power management calls in a multi-tasking environment (such as Windows) and identifies power-saving opportunities not apparent to applications. The application software communicates power-saving data via predefined APM interfaces. Windows 95 adopted APM to shut down the computer. It uses a special mode of the latest Intel processors – System Management Mode, or SMM. SMM lets the BIOS take control of the machine at any time and manage power to peripherals. A BIOS' APM support can't be circumvented by other software. This could cause a crash. Microsoft, Intel, Toshiba and others are now working on a new spec, called ACPI – Advanced Configuration and Power Interface. www.intel.com/IAL/power-mgm/apmovr.htm and www.ata.or/~acpi/.

APNIC Asia Pacific Network Information Center. A group formed to coordinate and promote TCP/IP based networks in the Asia-Pacific region. APNIC is responsible for management and assignment of IP (Internet Protocol) addresses in the Asia-Pacific, just as are ARIN and RIPE in the regions of the Americas and Europe, respectively. See also ARIN, IP, and RIPE.

APO Adaptive Performance Optimization. A technology used on the Texas Instruments ThunderLAN chipset, which was jointly developed by Compaq and Texas Instruments. APO dynamically adjusts critical parameters for minimum latency, minimum host CPU utilization and maximum system performance. This technology ensures that the capabilities of the PCI interface are used for automatically tuning the controller to the specific system in which it is operating.

Apocalypse, Four Horsemen Of The four horsemen of the Apocalypse were War, Plague, Famine and Death.

apogee The point on a satellite orbit that is most distant from the center of the gravitational field of the Earth. The point in an orbit at which the satellite is closest to the Earth is known as the perigee. In commercial application, the terms have most significance with respect to LEOs (Low Earth Orbiting) and MEOs (Middle Earth Orbiting) satellite constellations, which travel in elliptical orbits. See LEO and MEO.

apologize To lay the foundation for a future offense.

APON Originally specified by FSAN (Full Service Access Network) and subsequently standardized by the ITU-T as G.983.3, APON (ATM Passive Optical Network) is a local loop technology running the ATM protocol over single mode fiber. Synonymous with BPON (Broadband PON) APON runs at 155 Mbps or 622 Mbps downstream at a wavelength of 1490nm for voice and data and 1550nm for video transmission. The upstream speed is 155 Mbps at 1310nm for voice and data. The maximum logical reach of BPON is 20km, and the split ratio is 32:1. See also BPON, EPON, FSAN, GPON and PON.

APOT Additional Point Of Termination. The significance of APOT is that in the CLEC environment APOT is a requirement to submit LSR orders for collocation. These are some requirements that apply to APOT from Bell's point of view: APOT= Location "A" tie down information; CFA= Location "Z" tie down information; ACTL= Location "A" CLLI; LST= Location "Z" CLLI.

apparent power The mathematical product of the RMS current and the RMS voltage. Identical to the VA rating.

APPC Advanced Program-To-Program Communications. In SNA, the architectural component that allows sessions between peer-level application transaction programs. The LUs (Logical Units) that communicate during these sessions are known as LU type 6.2. APPC is an IBM protocol analogous to the OSI model's session layer: it sets up the necessary conditions that enable application programs to send data to each other through the network.

APPC/PC An IBM product that implements APPC on a PC.

appearance Usually refers to a private branch exchange line or extension which is on (i.e. "appears") on a multi-button key telephone. For example, extension 445 appears on three key systems.

appearance test point The point at which a circuit may be measured by test equipment.

append To add the contents of a list, or file, to those of another.

APPGEN A shortened form of the words APPlications GENerator.

Apple Computer, Inc. Cupertino, CA. manufacturer of personal computers. Heavy penetration in the graphics/desktop publishing business and in education. Apple was formed on April Fool's Day, 1976, by Steve Wozniak and Steve Jobs, aided greatly by Mike Markkula.

Apple Desktop Bus The interface on a Mac where non-peripheral devices, such as the keyboard, attaches. A Mac keyboard or mouse is called an ADB device. Contrast with peripherals, which attach through the SCSI interface. See also USB, which is a new bus for use on PCs but fulfilling essentially the same function as the Apple Desktop Bus.

Apple Desktop Interface ADI. A set of user-interface guidelines, developed by Apple Computer and published by Addison-Wesley, intended to ensure that the appearance and operation of all Macintosh applications are similar.

Apple Menu The Apple icon in the upper left hand corner of the Apple Macintosh screen. The Apple menu contains aliases, control panels, the chooser and other desk accessories.

Apple Pie Both an American icon, and the name chosen for Apple Computer's Personal Interactive Electronics (PIE) division, chartered with extending the company into new growth areas such as Personal Digital Assistants (PDAs), e.g. the Apple Newton. The PIE division includes Apple Online Services, Newton and Telecommunications group, publishing activities, and ScriptX-based multimedia PDA development.

Apple Remote Access ARA is Apple Computer's dial-in client software for Macintosh users allowing remote access to Apple and third party servers.

Apple URP Apple Update Routing Protocol. The network routing protocol developed by Apple for use with Appletalk.

AppleShare Apple Computer's local area network. It uses AppleTalk protocols. AppleShare is Apple system software that allows sharing of files and network services via a file server in the Apple Macintosh environment. See AppleTalk.

applet Mini-programs that can be downloaded quickly and used by any computer equipped with a Java-capable browser. Applets carry their own software players. See Java.

AppleTalk Apple Computer's proprietary networking protocol for linking Macintosh computers and peripherals, especially printers. This protocol is independent of what network it is layered on. Current implementations exist for LocalTalk (230.4 Kbps) and EtherTalk (10 Mbps).

AppleTalk Address Resolution Protocol See AARP.

AppleTalk Zone and Device Filtering Provides an additional level of security for AppleTalk networks. On AppleTalk networks, network managers can selectively hide or show devices and/or zones to ARA clients. See ARA.

appliance See Edge Appliance.

appliance creep Gadget creep in an enterprise network environment. For example, over time, various groups in the enterprise, including branch offices and remote sites, may install firewalls, intrusion detection systems, load-balancing devices, various types of WAN acceleration appliances, and other network devices, each of which performs a specific, narrow function. Each of these appliances also has power, interface, and space requirements, which create network management challenges. The figurative or literal string of appliances on a network is sometimes called an appliance conga line.

appliance conga line See appliance creep.

application A software program that carries out some useful task. Database managers, spreadsheets, communications packages, graphics programs and word processors are all applications.

application acceleration The use of one or more techniques by a WAN accelerator to improve perceived application response time across a WAN. These techniques include compression and coalescing.

application based call routing In addition to the traditional methods of routing and tracking calls by trunk and agent group, the latest Automatic Call Distributors route and track calls by application. An application is a type of call, for example, sales or service. Tracking calls in this manner allows accurately reported calls, especially when they are overflowed to different agent groups. See ACD.

Application Binary Interface ABI. The rules by which software code is written to operate specific computer hardware. Application software, written to conform to an ABI, is able to be run on a wide variety of system platforms that use the computer hardware for which the ABI is designed.

application bridge Aspect Telecommunications' ACD to host computer link. Originally it ran only over R2-232 serial connections, but it now runs over Ethernet, using the TCP/IP link protocol. See also Open Application Interface.

application class An SCSA term. A group of client applications that perform similar services, such as voice messaging or fax-back services.

Application Entity AE. A cellular radio term. An Application Entity provides the

EXHIBIT H

New Oxford American Dictionary

THIRD EDITION

Edited by

Angus Stevenson

Christine A. Lindberg

FIRST EDITION

Elizabeth J. Jewell

Frank Abate

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Oxford University Press, Inc., publishes works that further
Oxford University's objective of excellence
in research, scholarship, and education.

Oxford New York

Auckland Cape Town Dar es Salaam Hong Kong Karachi
Kuala Lumpur Madrid Melbourne Mexico City Nairobi
New Delhi Shanghai Taipei Toronto

With offices in

Argentina Austria Brazil Chile Czech Republic France Greece
Guatemala Hungary Italy Japan Poland Portugal Singapore
South Korea Switzerland Thailand Turkey Ukraine Vietnam

First edition 2001
Second edition 2005
Third edition 2010

Published by Oxford University Press, Inc.
198 Madison Avenue, New York, NY 10016
www.oup.com

Oxford is a registered trademark of Oxford University Press

The Library of Congress Cataloging-in-Publication Data
Data available

ISBN 978-0-19-539288-3

10

Printed in the United States of America
on acid-free paper

who's who ▶ n. a list or directory of facts about notable people.

wh-ques·tion ▶ n. a question in English introduced by a wh-word and requiring more information in reply than simply *yes* or *no*.

whump /wəmp/ ▶ n. [usu. in sing.] a dull thudding sound: *the horse fell with a great whump.*
▶ v. [no obj.] make such a sound: *he pitched a snowball that whumped into the car.* ■ [with obj.] strike (something) heavily with such a sound: *she began whumping him on his lower back.*
– ORIGIN late 19th cent.: imitative.

whup /(h)wəp/ ▶ v. (**whups, whupping, whupped**) [with obj.] informal beat; thrash: *they would whup him and send him home.* ■ defeat convincingly: *if you lined up our guys against the Eagers, they'd get whupped.*
– ORIGIN late 19th cent.: variant of WHIP.

wh-word ▶ n. Grammar any of a class of English words used to introduce questions and relative clauses. The main wh-words are *why, who, which, what, where, when,* and *how.*

why /(h)wī/ ▶ adv. for what reason or purpose: *why did he do it?* ■ [with negative] used to make or agree to a suggestion: *why don't I give you a lift?*
▶ relative adv. (with reference to a reason) on account of which; for which: *the reason why flu shots need repeating every year is that the virus changes.* ■ the reason for which: *each has faced similar hardships, and perhaps that is why they are friends.*
▶ exclam. **1** expressing surprise or indignation: *Why, that's absurd!*
2 used to add emphasis to a response: *"You think so?" "Why, yes."*
▶ n. (pl. **whys**) a reason or explanation: *the whys and wherefores of these procedures need to be explained to students.*
– PHRASES **why so?** for what reason or purpose?
– ORIGIN Old English *hwī, hwȳ* 'by what cause,' instrumental case of *hwæt* 'what,' of Germanic origin.

whyd·ah /'(h)widə/ (also **whyda**) ▶ n. an African weaverbird, the male of which has a black back and a very long black tail used in display flight. ● Genus *Vidua*, family Ploceidae: several species.
– ORIGIN late 18th cent. (originally *widow-bird*): alteration by association with *Whidah* (now Ouidah), a town in Benin.

WI ▶ abbr. ■ West Indies. ■ Wisconsin (in official postal use). ■ Brit. Women's Institute.

WIC /wik/ ▶ abbr. Women, Infants, and Children (a federal or state program to ensure proper nutrition for poor mothers and their children).

Wic·ca /'wikə/ ▶ n. the religious cult of modern witchcraft, esp. an initiatory tradition founded in England in the mid 20th century and claiming its origins in pre-Christian pagan religions.
– DERIVATIVES **Wic·can** adj. & n.
– ORIGIN representing Old English *wicca* 'witch.'

Wich·i·ta /'wiCHəˌtô, -ˌtă/ a city in southern Kansas, on the Arkansas River, the largest city in the state; pop. 366,046 (est. 2008).

Wich·i·ta Falls an industrial and commercial city in north central Texas; pop. 101,202 (est. 2008).

wick[1] /wik/ ▶ n. a strip of porous material up which liquid fuel is drawn by capillary action to the flame in a candle, lamp, or lighter. ■ Medicine a gauze strip inserted in a wound to drain it.
▶ v. [with obj.] absorb or draw off (liquid) by capillary action: *these excellent socks will wick away the sweat* | [no obj.] *synthetics with hollow fibers that wick well.*
– PHRASES **dip one's wick** vulgar slang (of a man) have sexual intercourse.
– ORIGIN Old English *wēoce*; related to Dutch *wiek* and German *Wieche* 'wick yarn.'

wick[2] ▶ n. **1** (in place names) a town, hamlet, or district: *Hampton Wick | Warwick.*
2 [in dialect] a dairy farm.
– ORIGIN Old English *wīc* 'dwelling place,' probably based on Latin *vicus* 'street, village.'

wick·ed /'wikid/ ▶ adj. (**wickeder, wickedest**) evil or morally wrong: *a wicked and unscrupulous politician.* ■ intended to or capable of harming someone or something: *he should be punished for wicked driving.* ■ informal extremely unpleasant: *the wicked sun, the wind outside was wicked.* ■ playfully mischievous: *Ben has a wicked sense of humor.* ■ informal excellent; wonderful: *Sophie makes wicked cakes.*
– DERIVATIVES **wick·ed·ly** adv.
– ORIGIN Middle English: probably from Old English *wicca* 'witch' + -ED[1].

WORD TRENDS See SICK[1].

wick·ed·ness /'wikidnis/ ▶ n. the quality of being evil or morally wrong: *the wickedness of the regime.*

wick·er /'wikər/ ▶ n. pliable twigs, typically of willow, plaited or woven to make items such as furniture and baskets: [as modifier] *a wicker chair.*
– ORIGIN Middle English: of Scandinavian origin; compare with Swedish *viker* 'willow'; related to *vika* 'to bend.'

wick·er·work /'wikərˌwərk/ ▶ n. wicker. ■ furniture or other items made of wicker.

wick·et /'wikit/ ▶ n. **1** (also **wicket door** or **wicket gate**) a small door or gate, esp. one beside or in a larger one. ■ an opening in a door or wall, often fitted with glass or a grille and used for selling tickets or a similar purpose. ■ one of the wire hoops on a croquet course.
2 Cricket each of the sets of three stumps with two bails across the top at either end of the pitch, defended by a batsman. ■ the prepared strip of ground between these two sets of stumps. ■ the dismissal of a batsman; each of ten dismissals regarded as marking a division of a side's innings: *Darlington won by four wickets.*
– PHRASES **a sticky wicket** Cricket a pitch that has been drying after rain and is difficult to bat on. ■ [in sing.] informal a tricky or awkward situation: *the problem of who sits where can create a sticky wicket.* **take a wicket** Cricket (of a bowler or a fielding side) dismiss a batsman.
– ORIGIN Middle English (in the sense 'small door or grille'): from Anglo-Norman French and Old Northern French *wiket*; origin uncertain, usually referred to the Germanic root of Old Norse *vīkja* 'to turn, move.' Cricket senses date from the late 17th cent.

wick·et-keep·er /'wikitˌkēpər/ ▶ n. Cricket a fielder stationed close behind a batsman's wicket and typically equipped with gloves and pads.
– DERIVATIVES **wick·et-keep·ing** n.

wick-i-up /'wikēˌəp/ ▶ n. an American Indian hut consisting of an oval frame covered with brushwood or grass.
– ORIGIN Fox 'house;' compare with WIGWAM.

Wick·low /'wiklō/ a county in eastern Republic of Ireland, in the province of Leinster. ■ its county town, on the Irish Sea; pop. 6,930 (2006).

wid·der·shins /'widərˌSHinz/ (also **withershins**) ▶ adv. chiefly Scottish in a direction contrary to the sun's course, considered as unlucky; counterclockwise.
– ORIGIN early 16th cent.: from Middle Low German *weddersins*, from Middle High German *widersinnes*, from *wider* 'against' + *sin* 'direction'; the second element was associated with Scots *sin* 'sun.'

wide /wīd/ ▶ adj. (**wider, widest**) **1** of great or more than average width: *a wide road.* ■ (after a measurement and in questions) from side to side: *it measures 15 cm long by 12 cm wide | how wide do you think this house is?* ■ open to the full extent: *wide eyes.* ■ considerable: *tax revenues have undershot Treasury projections by a wide margin.*
2 including a great variety of people or things: *a wide range of opinion.* ■ spread among a large number of people or over a large area: *the business is slowly gaining wider acceptance.* ■ [in combination] extending over the whole of: *an industry-wide trend.*
3 at a considerable or specified distance from a point or mark: *Bodie's shot was inches wide.* ■ Baseball (of a pitch) outside: *the ball was wide of the plate.* ■ Baseball (of a throw) to either side of a base: *forced a wide throw to first.* ■ (in field sports) at or near the side of the field: *he played in a wide left position.*
4 Phonetics another term for LAX.
▶ adv. **1** to the full extent: *his eyes opened wide.*
2 far from a particular point or mark: *a shot that went wide to the right.* ■ at or near the side of the field; toward the sideline: *he will play wide on the right.*
▶ n. Cricket a ball that is judged to be too wide of the stumps for the batsman to play, for which an extra is awarded to the batting side.
– PHRASES **give someone/something a wide berth** see BERTH. **wide awake** fully awake. **wide of the mark** a long way away from an intended target. ■ inaccurate: *the accusation was a little wide of the mark.* **wide open 1** fully open: *the door was wide open.* **2** (of an issue or a contest) completely unresolved or unpredictable. **3** vulnerable, esp. to attack.
– DERIVATIVES **wide·ness** n., **wid·ish** adj.
– ORIGIN Old English *wīd* 'spacious, extensive,' *wīde* 'over a large area,' of Germanic origin.

wide-an·gle ▶ adj. (of a lens) having a short focal length and hence a field covering a wide angle.

wide ar·e·a net·work (abbr.: **WAN**) ▶ n. a computer network in which the computers connected may be far apart, generally having a radius of half a mile or more. Compare with LOCAL AREA NETWORK.

wide-a-wake ▶ n. a soft felt hat with a low crown and wide brim.
– ORIGIN mid 19th cent.: punningly so named, because the hat does not have a nap.

wide-band ▶ adj. (of a radio, or other device or activity involving broadcasting) having or using a wide band of frequencies or wavelengths.

wide-bod·y /'wīd ˌbädē/ ▶ adj. [attrib.] (also **wide-bodied**) having a wide body, in particular: ■ (of a large jet airplane) having a wide fuselage. ■ (of a tennis racket) having a wide head.
▶ n. (pl. **wide-bodies**) (also **widebody**) **1** a large jet airplane with a wide fuselage.
2 a tennis racket with a wide head.
3 informal a large, heavily built person, esp. one who plays a team sport.

wide-eyed ▶ adj. having one's eyes wide open in amazement. ■ innocent: *a wide-eyed country boy.*

wide·ly /'wīdlē/ ▶ adv. **1** over a wide area or at a wide interval: *he smiled widely and held out a hand | a tall man with widely spaced eyes.* ■ to a large degree in nature or character (used to describe considerable variation or difference): *lending policies vary widely between different banks* | [as submodifier] *people in widely different circumstances.*
2 over a large area or range; extensively: *Deborah has traveled widely* | [as submodifier] *she was widely read.* ■ by many people or in many places: *credit cards are widely accepted.*

wid·en /'wīdn/ ▶ v. make or become wider: [with obj.] *the incentive to dredge and widen the river* | [no obj.] *his grin widened | the lane widened out into a small clearing.*
– DERIVATIVES **wid·en·er** n.

wide-out /'wīdˌout/ ▶ n. a wide receiver.

wide-rang·ing ▶ adj. covering an extensive range: *a wide-ranging discussion.*

wide re·ceiv·er ▶ n. Football an offensive player who is positioned at a distance from the end and is used primarily as a pass receiver.

wide·screen /'wīdˌskrēn/ (also **widescreen**) ▶ adj. [attrib.] designed with or for a screen presenting a wide field of vision in relation to its height: *a widescreen TV.*
▶ n. a movie or television screen presenting a wide field of vision in relation to its height. ■ a film format presenting a wide field of vision in relation to height.

wide·spread /'wīdˈspred/ ▶ adj. found or distributed over a large area or number of people: *there was widespread support for the war.*

widg·eon ▶ n. variant spelling of WIGEON.

widg·et /'wijit/ ▶ n. informal a small gadget or mechanical device, esp. one whose name is unknown or unspecified. ■ Computing an application, or a component of an interface, that enables a user to perform a function or access a service.
– ORIGIN 1930s: perhaps an alteration of GADGET.

wid·ow /'widō/ ▶ n. **1** a woman who has lost her husband by death and has not remarried. ■ [with modifier] humorous a woman whose husband is often away participating in a specified sport or activity: *a golf widow.*
2 Printing a last word or short last line of a paragraph falling at the top of a page or column and considered undesirable.
▶ v. [with obj.] (usu. as adj. **widowed**) make into a widow or widower: *she had to care for her widowed mother.*
– ORIGIN Old English *widewe*, from an Indo-European root meaning 'be empty'; compare with Sanskrit *vidh* 'be destitute,' Latin *viduus* 'bereft, widowed,' and Greek *ēitheos* 'unmarried man.'

wid·ow·er /'widō-ər/ ▶ n. a man who has lost his wife by death and has not remarried.

wid·ow·hood /'widōˌhŏŏd/ ▶ n. the state or period of being a widow or widower.

wid·ow-mak·er ▶ n. informal a thing with the potential to kill men. ■ a dead branch caught precariously high in a tree which may fall on a person below.

wid·ow's mite ▶ n. a small monetary contribution from someone who is poor.
– ORIGIN with biblical allusion to Mark 12:43.

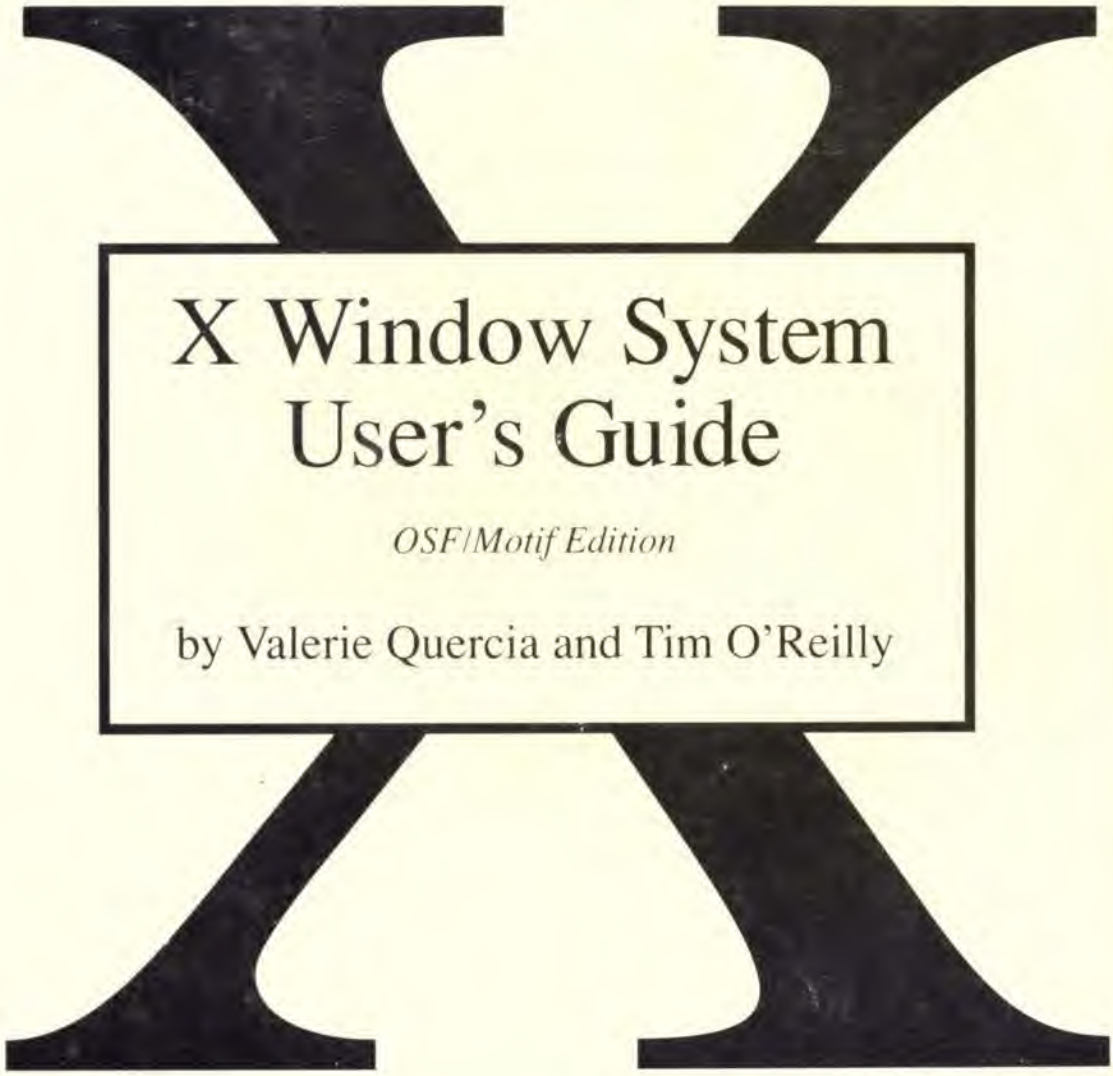
wid·ow's peak ▶ n. a V-shaped growth of hair toward the center of the forehead, esp. one left by a receding hairline in a man.
– ORIGIN mid 19th cent.: so called because it was formerly believed to be a predictor of widowhood for a woman.

EXHIBIT I

The Definitive Guides to the X Window System

Volume Three

Motif Edition



X Window System User's Guide

OSF/Motif Edition

by Valerie Quercia and Tim O'Reilly

O'Reilly & Associates, Inc.

X Window System User's Guide

The X Window System

The books in the X Window System Series are based in part on the original MIT X Window System documentation, but are far more comprehensive, easy to use, and are loaded with examples, tutorials and helpful hints. Over 20 major computer vendors recommend or license volumes in the series. In short, these are the definitive guides to the X Window System.

Volume 0:

X Protocol Reference Manual

A complete programmer's reference to the X Network Protocol, the language of communication between the X server and the X clients, R4. 500 pages. $30.00.

Volumes 1 and 2:

Xlib Programming Manual
Xlib Reference Manual

Revised for Release 4. Complete guide and reference to programming with the X library (Xlib), the lowest level of programming interface to X. 672 and 792 pages. $60.00 for the set, or $34.95 each.

Volume 3:

X Window System User's Guide (Standard and Motif Editions)

Revised and enlarged to include X11 Release 4. Describes window system concepts and the most common client applications available for X11. Includes complete explanation of window managers. For experienced users, later chapters explain customizing the X environment. Standard Edition shows *the twm* window manager. 752 pages. $30.00. Motif Edition highlights *mwm*. 610 pages. $30.00.

Volumes 4 and 5:

X Toolkit Intrinsics Programming Manual (Standard and Motif Editions)
X Toolkit Intrinsics Reference Manual

Complete guides to programming with the X Toolkit. The *Programming Manual* provides concepts and examples for using widgets and for the more complex task of writing new widgets. The Standard Edition shows Athena widgets. 624 pages. $30.00. The Motif Edition shows Motif widget examples. 612 pages. $30.00. The *Reference Manual* provides reference pages for Xt functions, and Xt and Athena widgets. 776 pages. $30.00.

Volume 7:

XView Programming Manual

XView is an easy-to-use toolkit that is not just for Sun developers. It is available on MIT's R4 tape and System V Release 4, as well as being a part of Sun's Open Windows package. This manual provides complete information on XView, from concepts to creating applications to reference pages. 672 pages. $30.00.

The X Window System in a Nutshell

For the experienced X programmer, contains essential information from other volumes of the series in a boiled-down, quick-reference format that makes it easy to find the answers needed most often. 380 pages. $24.95.

For orders or a free catalog of all our books, please contact us.

O'Reilly & Associates, Inc.

Books That Help People Get More Out of Computers

632 Petaluma Avenue, Sebastopol, CA 95472

email: uunet!oralnuts · (800) 338-6887 · (707) 829-0515 · FAX (707) 829-0104

Volume Three

X Window System User's Guide

OSF/Motif Edition

by Valerie Quercia and Tim O'Reilly

O'Reilly & Associates, Inc.

Printing and Revision History

Sept. 1988:	First Edition
July 1989:	Second Edition. Revised to reflect Release 3.
Oct. 1989:	Minor corrections.
May 1990:	Third Edition. Revised to reflect Release 4.
Jan. 1991:	Motif Edition.

Small Print

ISBN 0-937175-61-7

Table of Contents

Preface

By convention, a preface describes the book itself, while the introduction describes the subject matter. You should read through the preface to get an idea of how the book is organized, the conventions it follows, and so on.

In The Preface:

Preface

The X Window System™ is a network-based graphics windowing system. It was developed by MIT and has been adopted as an industry standard. X provides the bare bones of a window system upon which almost any style of graphical user interface (GUI) can be built. One of the most popular user interfaces available in the market today is OSF/Motif™ developed by the Open Software Foundation.™

The *X Window System User's Guide, Motif Edition* describes window system concepts, the application programs (clients) commonly distributed with Version 11, Release 4 of X, and how you can expect programs to operate with OSF/Motif. (The *Motif Edition* is intended for those using X with the OSF/Motif interface. Another edition of the *X Window System User's Guide* is available for those using X without Motif.)

Assumptions

This book assumes that X is already installed on your system and that all standard MIT clients are available. In addition, although X runs on many different systems, this book assumes that you are running it on a UNIX® system and that you have basic familiarity with UNIX. If you are not using UNIX, you will still find this book useful—UNIX dependencies are not that widespread—but you may occasionally need to translate a command example into its equivalent on your system. This book also assumes that you are using a three-button pointer (e.g., a mouse) and that the operation of the *mwm* window manager is controlled by the *system.mwmrc* file provided with OSF/Motif 1.1. (If this is not the case, the book provides information that will allow you to understand how *mwm* is configured on your system.)

This book is written for both first-time and experienced users of the X Window System. First-time users should read the book in order, starting with Chapter 1, *An Introduction to the X Window System*.

Experienced users can use this book as a reference for the client programs detailed here. Since there is great flexibility with X, even frequent users need to check on the syntax and availability of options. Reference pages for each client detail command line options, customization database (resource database) variables, and other detailed information.

The book contains these parts:

Part One: Using X

Preface
Describes the book's assumptions, audience, organization, and conventions.

Chapter 1: An Introduction to the X Window System
Describes the basic terminology associated with the X Window System: server, client, window, etc. The most important X clients are described.

Chapter 2: Getting Started
Shows how to start the programs necessary to begin using X: the server, the first terminal window, and window manager. This chapter is tutorial in nature: you can follow along at a workstation as you read.

Chapter 3: Working in the X Environment
Teaches the skills necessary to begin working productively. Shows you how to add additional windows; exit from a window; use the tools of the display; perform some basic window manager operations using the *mwm* window manager; set up the display.

Introduces two methods of customizing X client programs: command line options and resource variables.

Compares a standard X application (created using the X Toolkit) with a Motif application (created using the Motif Toolkit).

Chapter 4: More about the mwm Window Manager
Describes additional window manager operations, such as resizing windows and changing the order of windows in the stack.

Chapter 5: The xterm Terminal Emulator
Describes how to use the *xterm* terminal emulator, the most frequently used client. Certain aspects of *xterm* operation described in this chapter, such as scrolling and "copy and paste," are common to other applications as well.

Chapter 6: Font Specification
Describes the somewhat complicated font naming conventions and ways to simplify font specification, including wildcarding and aliasing. Describes how to use the *xlsfonts*, *xfd*, and *xfontsel* clients to list, display, and select available display fonts.

Chapter 7: Graphics Utilities
Explains how to use the major graphics clients included with X, notably the *bitmap* editor.

Chapter 8: Other Clients
Gives an overview of other clients available with X, including window and display information clients, the *xkill* program, and several "desk accessories."

In describing various standard clients, this chapter highlights those features provided by the Athena widget set. This chapter also describes some of the features of applications written using the Motif widget set.

Part Two: Customizing X

Chapter 9: Command Line Options
Discusses some of the standard command line options accepted by most clients.

Chapter 10: Setting Resources
Tells how to create an *.Xresources* file to set default characteristics for client applications. This chapter also describes how to use *xrdb*, which saves you from having to maintain multiple *.Xresources* files if you run clients on multiple machines.

Chapter 11: Customizing mwm
Describes the *.mwmrc* file by showing the default file shipped by OSF and examines the purpose and syntax of entries. Explains various techniques for revising the *.mwmrc* file to modify existing menus and create new ones. Reviews the different types of resources that can be used to control *mwm*.

Chapter 12: Setup Clients
Describes how to set display and keyboard preferences using *xset* and how to set root window preferences using *xsetroot*. Also demonstrates how to use *xmodmap* to redefine the logical keynames and pointer commands recognized by X.

Part Three: Client Reference Pages

Extended reference pages for all clients.

Part Four: Appendices

Appendix A: System Management

Appendix B: Release 4 Standard Fonts

Appendix C: Standard Bitmaps

Appendix D: Standard Cursors

Appendix E: xterm Control Sequences

Appendix F: Translation Table Syntax

Appendix G: Athena Widget Resources

Glossary

Index

Bulk Sales Information

This guide is being resold by many workstation manufacturers as their official X Window System documentation. For information on volume discounts for bulk purchases, call O'Reilly & Associates, Inc., at 800-338-6887, or send email to linda@ora.com (uunet!ora!linda).

For companies requiring extensive customization of the guide, source-licensing terms are also available.

xshowfonts.c and Other Free Programs

The source to *xshowfonts.c*, which is printed in Appendix B, *Release 4 Standard Fonts*, is also available free from UUNET (that is, free except for UUNET's usual connect-time charges). If you have access to UUNET, you can retrieve the source code using UUCP or FTP. For UUCP, find a machine with direct access to UUNET, and type the following command:

```
uucp uunet\!~uucp/nutshell/Xuser/xshowfonts.c.Z yourhost\!~/yourname/
```

The backslashes can be omitted if you use the Bourne shell (*sh*) instead of *csh*. The file should appear some time later (up to a day or more) in the directory */usr/spool/uucppublic/yourname*.

You don't need to subscribe to UUNET to be able to use their archives via UUCP. By calling 1-900-468-7727 and using the login "uucp" with no password, anyone may *uucp* any of UUNET's online source collection. (Start by copying *uunet!~/ls-lR.Z*, which is a compressed index of every file in the archives. Peruse this file to get an idea of what sorts of programs are available. The file *uunet!~nutshell/ls-lR.Z* contains a listing of the files contained in the nutshell subdirectory (example programs for our books.)) As of this writing, the cost is 40 cents per minute. The charges will appear on your next telephone bill.

To use FTP, you will need to find a machine with direct access to the Internet. The following example is a sample session, with commands in boldface.

```
% ftp uunet.uu.net
Connected to uunet.uu.net.
220 uunet FTP server (Version 5.99 Wed May 23 14:40:19 EDT 1990) ready.
Name (uunet.uu.net:ambar): anonymous
331 Guest login ok, send ident as password.
Password: ambar@ora.com (use your user name and host here)
230 Guest login ok, access restrictions apply.
ftp> cd nutshell/Xuser
250 CWD command successful.
ftp> binary (you must specify binary transfer for compressed files)
200 Type set to I.
ftp> get xshowfonts.c.Z
200 PORT command successful.
150 Opening ASCII mode data connection for xshowfonts.c.Z (5587 bytes).
226 Transfer complete.
ftp> quit
221 Goodbye.
%
```

The file is a compressed C program. To uncompress it, type:

```
% uncompress xshowfonts.c
```

Acknowledgements

The first edition of this guide was based in part on three previous X Window System user's guides: one from Masscomp, which was written by Jeff Graber; one from Sequent Computer Systems, Inc., and one from Graphic Software Systems, Inc.; both of which were written by Candis Condo (supported by the UNIX development group). Some of Jeff's and Candis's material in turn was based on material developed under the auspices of Project Athena at MIT.

Most of the reference pages in Part Three of this guide have been adapted from reference pages copyright © 1990 the Massachusetts Institute of Technology, or from reference pages produced by Graphic Software Systems. Refer to the "Authors" section at the end of each reference page for details. Other copyrights are listed on the relevant reference pages.

Permission to use this material is gratefully acknowledged.

This guide was primarily developed using the MIT sample server on a Sun-3™ Workstation, with additional testing done on a Sony NEWS™ workstation running Sony's X implementation, a Visual 640 X Display Station™, and an NCD16™ Network Display Station.

We are grateful to Sony Microsystems for the loan of a Sony NEWS workstation and to Visual Technology Incorporated for the loan of a Visual 640 X Display Station. We appreciate the support of these manufacturers in helping us develop complete and accurate X Window System documentation.

We'd also like to thank the Open Software Foundation for permission to reprint the *system.mwmrc* file in Chapter 11, *Customizing mwm*. Special thanks to Elizabeth Connelly of OSF for arranging this.

Special thanks also to Dave Curry for his expert technical and editorial support.

Of course, the authors would like to thank the entire staff of O'Reilly and Associates for producing the book, and the staff of Cambridge Computer Associates, Inc., for lending their support.

Despite the efforts of these people, the standard authors' disclaimer applies: any errors that remain are our own.

Font and Character Conventions

These typographic conventions are used in this book:

Italics are used for:

- new terms where they are defined.
- file and directory names, and command and client names when they appear in the body of a paragraph.

`Courier` is used within the body of the text to show:

- command lines or options that should be typed verbatim on the screen.

is used within examples to show:

- computer-generated output.
- the contents of files.

`Courier bold` is used within examples to show command lines and options that should be typed verbatim on the screen.

`Courier italics` are used within examples or explanations of command syntax to show a parameter to a command that requres context-dependent substitution (such as a variable). For example, *`filename`* means to use some appropriate filename; *`option(s)`* means to use some appropriate option(s) to the command.

Helvetica is used to show menu titles and options.

These symbols are used within the *X Window System User's Guide*:

[ ] surround an optional field in a command line or file entry.

$ is the standard prompt from the Bourne shell, *sh*(1).

% is the standard prompt from the C shell, *csh*(1).

name(1) is a reference to a command called *name* in Section 1 of the *UNIX Reference Manual* (which may have a different name depending on the version of UNIX you use).

Part One:

Using X

Part One provides an overview of the X Window System and concepts, and describes how to use the most important programs available in the X environment.

1

An Introduction to the X Window System

This chapter describes the features of a typical X display, while introducing some basic window system concepts. It also provides an overview of the X Window System's client-server architecture and briefly describes the most commonly used clients.

In This Chapter:

1 An Introduction to the X Window System

The X Window System, called X for short, is a network-based graphics window system that was developed at MIT in 1984. Several versions of X have been developed, the most recent of which is X Version 11 (X11), first released in 1987.

X11 has been adopted as an industry standard windowing system. X is supported by a consortium of industry leaders such as DEC, Hewlett-Packard, Sun, IBM, and AT&T that have united to direct, contribute to, and fund its continuing development. In addition to the system software development directed by the X Consortium, many independent developers are producing application software specifically for use with X, including spreadsheets, database programs, and publishing applications.

First, we'll take a look at a typical X display and consider some general system features. We'll also briefly compare a standard X application (written with the X Toolkit) to a Motif application (written with the Motif Toolkit). Then we'll discuss what distinguishes the X Window System from other window systems. We'll also introduce some of the more important programs included in the standard distribution of X, and the *mwm* window manager shipped with OSF/Motif.

Anatomy of an X Display

X is typically run on a workstation with a large screen (although it also runs on PCs and special X terminals, as well as on many larger systems). X allows you to work with multiple programs simultaneously, each in a separate *window*. The display in Figure 1-1 includes five windows.

The operations performed within a window can vary greatly, depending on the type of program running it. Certain windows accept input from the user: they may function as terminals, allow you to create graphics, etc. Other windows simply display information, such as the time of day or a picture of the characters in a particular font, etc.

The windows you will probably use most frequently are *terminal emulators*, windows that function as standard terminals. The terminal emulator included with the standard release of X is called *xterm*. Figure 1-1 depicts three *xterm* windows. In an *xterm* window, you can do anything you might do in a regular terminal: enter commands, run editing sessions, compile programs, etc.

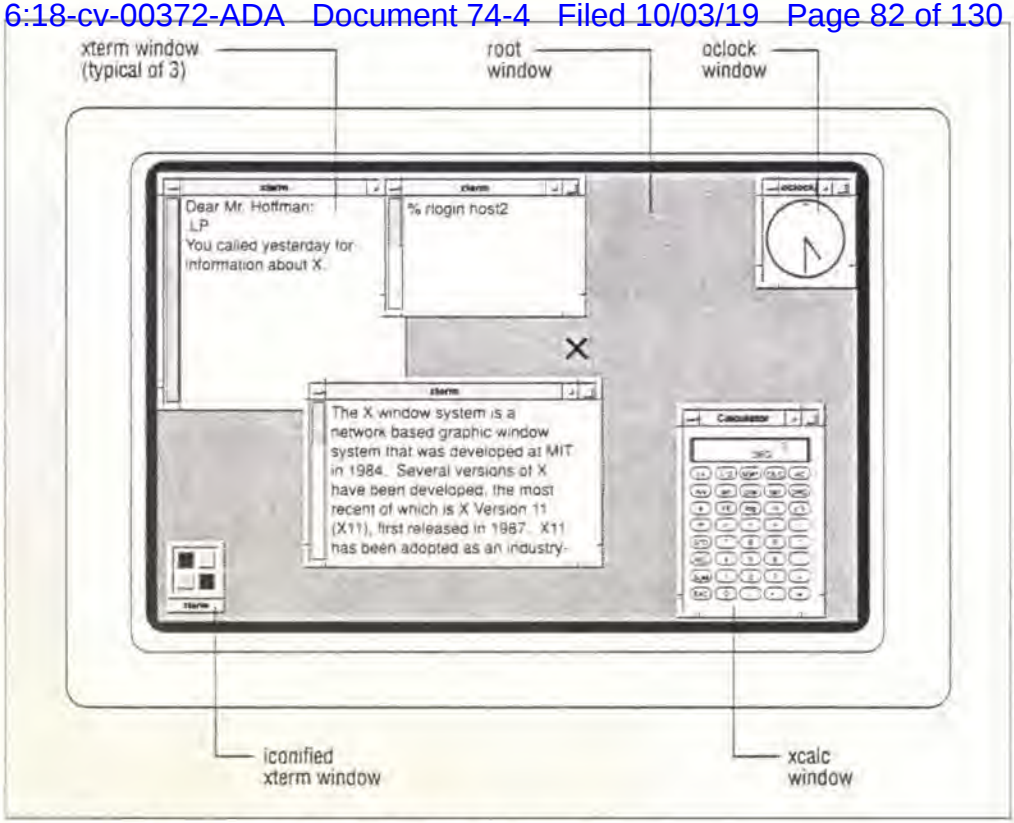


Figure 1-1. X display with five windows and an icon

The display in Figure 1-1 also includes two other application windows: a clock (called *xclock*) and a calculator (*xcalc*). X provides many such small utility programs—analogous to the so-called "desk accessories" of the Macintosh environment—intended to make your work easier.

The shaded area that fills the entire screen is called the *root* (or *background*) window. Application windows are displayed on top of this root window. X considers windows to be related to one another in a hierarchy, similar to a family tree. The root window is the root or origin within this hierarchy and is considered to be the *parent* of application windows displayed on it. Conversely, these application windows are called *children* of the root window. In Figure 1-1, the *xterm*, *xclock*, and *xcalc* windows are children of the root window.

As we'll see later, the window hierarchy is actually much more complicated than this "two generation" model suggests. Various *parts* of application windows are windows in their own right. For example, many applications provide menus to assist the user. Technically speaking, these menus are separate windows. Knowledge of the complexity of the window hierarchy (and the composite parts of an application) will become important when we discuss how to tailor an application to better suit your needs.

One of the strengths of a window system such as X is that you can have several processes going on simultaneously in several different windows. For example, in Figure 1-1 the user is logging in to a remote system in one *xterm* window and editing a text file in each of the two other *xterm* windows. (As we'll see in Chapter 5, *The xterm Terminal Emulator*, you can also cut and paste text between two windows.) Be aware, however, that you can only input to one window at a time.

Another strength of X is that it allows you to run programs on machines connected by a network. You can run a process on a remote machine while displaying the results on your own screen. You might want to access a remote machine for any number of reasons: to use a program or access information not available on your local system; to distribute the work load, etc. We'll discuss X's networking capabilities in more detail in the "X Architecture Overview" later in this chapter.

Now let's take another look at our sample display in Figure 1-1. Notice that the *xterm* windows overlap each other. Windows often overlap much like sheets of paper on your desk or a stack of cards. Be aware that overlapping does not interfere with the process run in each window. However, in order to really take advantage of windowing, you need to be able to move and modify the windows on your display. For example, if you want to work in a window that is partially covered by another window, you need to be able to raise the obscured window to the top of the window stack.

Window management functions are provided by a type of program called a *window manager*. The window manager controls the general operation of the window system, allowing you to change the size and position of windows on the display. You can reshuffle windows in a window stack, make windows larger or smaller, move them to other locations on the screen, etc. The window manager provides part of the user interface—to a large extent it controls the "look and feel" of the X Window System.

The window manager provided with OSF/Motif is called *mwm*, the Motif window manager. The most distinguishing feature of *mwm* is the "frame" it places around all windows on the display. Notice that each top-level application window in the illustration is surrounded by this frame. As we'll see, by clicking a mouse or other pointing device on various parts of the window frame, you can perform management functions on the window.

The *mwm* window frame is a composite of several parts, the most prominent of which are shown in Figure 1-2.

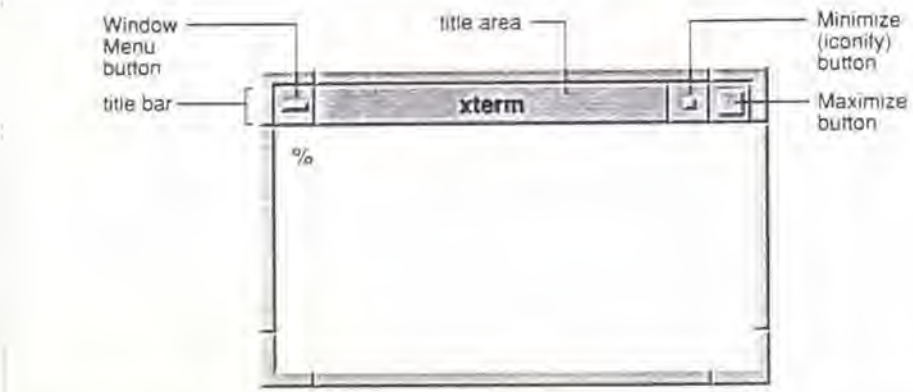


Figure 1-2. mwm frames each window on the display

The top edge of the frame is wider than the other three edges and features most of the window management tools. This wide horizontal bar spanning the top of the window is known as a *titlebar*. The large central portion of this top edge is called the *title area* mainly because it contains a text description of the window. (Generally, this is the application name, but as we'll see later, you can often specify an alternate title.) The titlebar also features three command buttons whose functionality we'll discuss in Chapter 3 and Chapter 4. We'll also see how to use the sides and bottom of the frame to resize a window and to raise it to the top of the window stack.

mwm attempts to create a three-dimensional appearance, which is somewhat more aesthetic than the look provided by many other window managers. You'll probably notice that window frames, various command buttons, icons, etc., appear to be raised to varying heights above screen level. This illusion is created by subtle shading and gives many display features a "beveled" look, similar to the beveled style of some mirrors.

mwm is intended to be used with the OSF/Motif graphical user interface. For those not using Motif, there are several other window managers available in the market today. In the standard distribution of X from MIT (as of Release 4), the official window manager is called *twm*. (*twm* originally stood for "Tom's window manager," in honor of its developer, Tom LaStrange. However, it has since been renamed the "tab window manager.") The *twm* window manager provides a different "look and feel" than *mwm*. Rather than framing application windows, *twm* simply provides each window with a titlebar, different from the *mwm* titlebar in style, but offering similar window management functions. Earlier releases of X supported a third window manager, *uwm*, the "universal window manager," which is still available as part of the user contributed part of the MIT X distribution. The contributed distribution includes several other window managers and still others are available commercially.

Aesthetics notwithstanding, one of the primary advantages *mwm* has over other window managers is inherent in the nature of a frame: it provides window management tools on four sides of the window. *twm*'s titlebar is a useful window management tool, but a titlebar is

often covered by other windows in the stack. In most cases at least a part of a window's frame should be visible and thus accessible to the user.

Also pictured in Figure 1-1 is an *icon*. An icon is a small symbol that represents a window in an inactive state. The window manager allows you to convert windows to icons and icons back to windows. You may want to convert a window to an icon to save space on the display or to prevent input to that window. Each icon has a label, generally the name of the program that created the window. The icon in Figure 1-1 represents a fourth *xterm* window on the display. Icons can be moved around on the display, just like windows.

The contents of a window are not visible when the window has been converted to an icon but they are not lost. In fact, a client continues to run when its window has been iconified; if you iconify a terminal emulator client such as *xterm*, any programs running in the shell will also continue.

If you've used other window managers, you may notice that icon symbols generated by *mwm* are somewhat larger and more decorated. The detail on icons is another aesthetic advantage of *mwm*.

All X displays require you to have some sort of pointer, often a three-button mouse, with which you communicate information to the system. As you slide the pointer around on your desktop, a cursor symbol on the display follows the pointer's movement. For our purposes, we will refer to both the pointing device (e.g., a mouse) and the symbol that represents its location on the screen as pointers. Depending on where the pointer is on the screen (in an *xterm* window, in another application window, on the root window, etc.), it is represented by a variety of cursor symbols. If the pointer is positioned on the root window, it is generally represented by an X-shaped cursor, as in Figure 1-1. If the pointer is in an *xterm* window, it looks like an uppercase I and is commonly called an *I-beam cursor*.*

A complete list of standard X cursors is shown in Appendix D, *Standard Cursors*. OSF/Motif provides some additional cursors. Some of the most common standard cursor shapes, as well as two Motif-specific cursors, are shown in Figure 1-3. As we'll see later, some applications allow you to select the cursor to use.

*Even though the actual image on the screen is called a cursor, throughout this guide we refer to "moving the pointer" to avoid confusion with the standard text cursor that can appear in an *xterm* window.

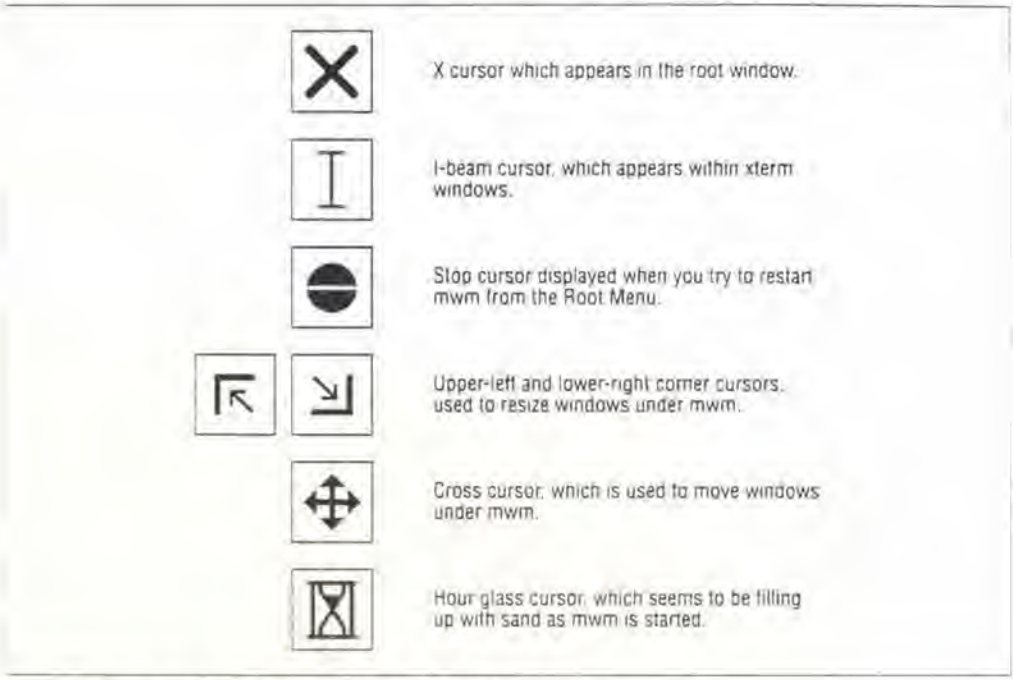


Figure 1-3. Some standard cursors and two Motif-specific cursors

You use the pointer to manage windows and icons, to make selections in menus, and to select the window in which you want to input. You can't type in an *xterm* window until you select that window using the pointer. Directing input to a particular window is called *focusing*. When a window has the input focus, the window frame and the text cursor (if any) are highlighted, as in Figure 1-4. The window to which input is directed is often called the *active* window.

Be aware that the frame may be highlighted in different ways, depending on the version of *mwm* you are running and the color resources specified for your system. The frame may change from black to white, from grey to white, etc. In any case, the active window's frame will be a different color than the frames of all other windows on the display.

Most window managers require you to select the active window in one of two ways: either by moving the pointer so that it rests within the desired window or by clicking the pointer on the window. By default, the Motif window manager requires you to click the pointer on the window to which you want to direct input. This focusing style is commonly referred to as "click-to-type" or ("explicit focus"). However, as we'll see in Chapter 11, *Customizing mwm*, *mwm* can be customized to allow you to direct input focus simply by moving the pointer. This focusing style is commonly referred to as "real-estate-driven" (or "pointer focus").

The *twm* window manager uses the real-estate-driven style: you direct input focus by moving the pointer into the desired window and leaving it there. As long as the pointer remains within the window's borders, the keystrokes you type will appear in that window (when the

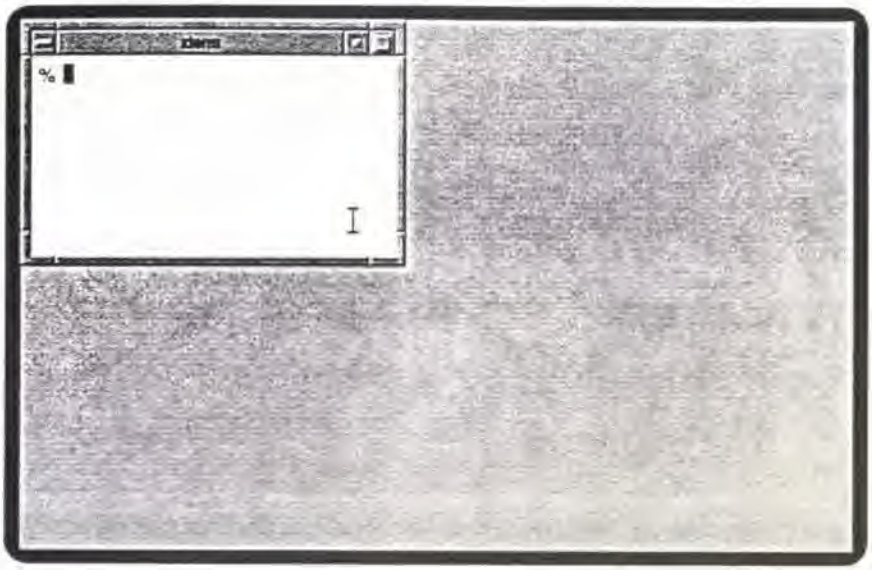


Figure 1-4. Focus on an xterm window

application accepts text input) or will somehow affect its operation (perhaps serve as commands). If you accidentally knock the pointer outside the window's borders, the keystrokes you type will not appear in that window or affect its operation. If you inadvertently move the pointer into another window, that window becomes the focus window. If you move the pointer onto the root window, the keystrokes are in effect lost—no application will interpret them.*

As previously mentioned, by default the Motif window manager uses the click-to-type focusing style: you must click the pointer on a window to focus input on that window. When you begin using X with *mwm*, you'll need to select the window to receive input by placing the pointer anywhere within the window and clicking the first (generally the leftmost) button. Once you select the focus window in this way, all input is directed to that window until you move the pointer and deliberately click on another window.

*In a few cases, the window manager may interpret these keystrokes. For example, you can customize *mwm* to display a menu when you type certain keystrokes while the pointer rests on the root window. See Chapter 11, *Customizing mwm*, for more information about mapping window manager functions to certain keys and pointer actions.

One of Motif's greatest strengths is that it allows you to choose the focus policy. This flexibility makes *mwm* a desirable choice for users with a variety of needs and work habits. As you might imagine, both focusing policies have their advantages. Click-to-type focus requires a little more work than pointer focus. (It's simpler to move the pointer than to move and click.) On the other hand, click-to-type focus is more precise—you can't inadvertently change the focus by moving the pointer.

We find click-to-type focus somewhat laborious. However, a touch typist, who is not inclined to look at the screen, might consider pointer focus too risky. It's possible to knock the pointer out of a window and lose a large amount of text before noticing. Another disadvantage of pointer focus is that it sometimes takes a moment for the input focus to catch up with the pointer, especially on slower machines. If you type right away, some keystrokes may end up in the window you left rather than in the new window. This is actually a bug that happens because of the additional overhead involved in complex window managers such as *mwm* or *twm*. Since you can change the focus policy rather simply, you might want to experiment with both methods. For now, we'll assume you're using the default click-to-type focus.

The most important thing to recognize is that focusing on the proper window is integral to working with an application running with a window system. If something doesn't work the way you expect, make sure that the focus is directed to the correct window. After you use X for a while, awareness of input focus will come naturally.

The pointer is also often used to display menus. Some X programs, notably *mwm* and *xterm*, have menus that are displayed by keystrokes and/or pointer button motions. More versatile than many other window managers, *mwm* provides two default menus, each representing a different menu "style."

The Window Menu is a "pull-down" menu that can be displayed on any window by placing the pointer on the small rectangular button in the upper-left corner of the frame and either clicking or pressing and holding down the first pointer button. Roughly defined, a pull-down menu is accessed from a graphical element that is always displayed, such as a command button or a menu bar. Figure 1-5 shows an *xterm* window with the Window Menu displayed by clicking the first pointer button in the menu button on the frame.

As you might infer from some of the menu items, you can use the Window Menu to move, resize, and otherwise manage the window on which it is displayed. When you display the Window Menu by clicking the first pointer button (as opposed to pressing and holding it down), the first the first item available for selection is surrounded by a box. In this case, the first available selection is Move. The first item on the menu, Restore, is used to change an icon back into a window; therefore, it is not useful at this time. The fact that Restore is not selectable is indicated by the fact that it appears in a lighter typeface. The Window Menu is discussed in detail in Chapter 4, *More about the mwm Window Manager*.

mwm also supports "pop-up" menus, which are displayed at the current pointer position. (Many X clients also use pop-up menus.) In addition to keyboard keys and pointer button motions, the location of the pointer plays a role in displaying menus. For example, *xterm* menus can only be displayed when the pointer is within an *xterm* window. Figure 1-6 shows the *mwm* Root Menu, which is generally displayed by placing the pointer on the root window and pressing and holding down the first pointer button.

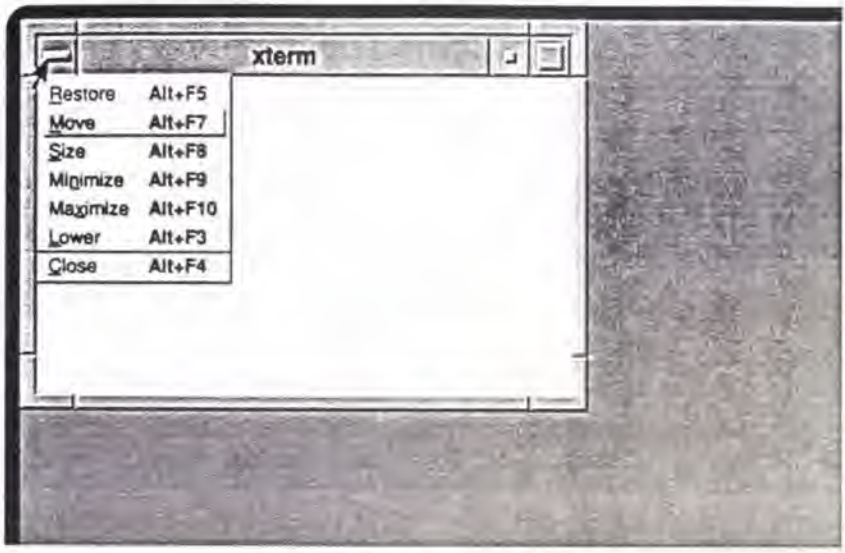


Figure 1-5. A pull-down menu: mwm's Window Menu

In Figure 1-6, the arrow next to the menu title represents the pointer. As you drag the pointer down the menu, each of the menu selections is highlighted. Regardless of the program, you generally select a menu item by dragging the pointer down the menu, highlighting the item you want, and releasing (or sometimes clicking) the pointer button. (*mwm* generally highlights an item by placing a rectangular box around it.) The Root Menu provides commands that can be thought of as affecting the entire display (as opposed to a single window). For example, the first menu item, New Window, creates a new *xterm* window on the local machine and display.

Though mwm's menus can be useful, you'll probably find that you perform most window management functions simply by using the pointer on the window frame. In Chapter 3, *Working in the X Environment*, we'll describe several of these functions. Keep in mind, however, that both of the menus can be useful in certain circumstances. For instance, the Window Menu may be useful when parts of the window frame are obscured by another window. The Root Menu can be customized to execute system commands, such as the *xterm* command initialized by the New Window menu item. It's fairly simple to add items to the Root Menu; you might want to add menu items to start some of the applications you use regularly. Chapter 11, *Customizing mwm*, describes how to add menu items and to perform a variety of modifications.

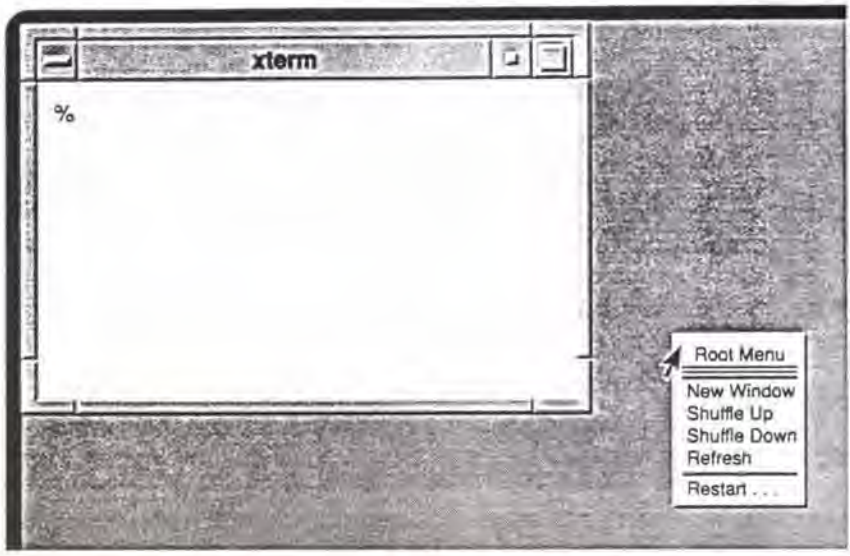


Figure 1-6. A pop-up menu: mwm's Root Menu

As we'll see in Chapter 8, *Other Clients*, some programs provide menus that you can display simply by placing the pointer on a particular part of the window, e.g., a horizontal menu bar across the top.

A final note about the X display: in X, the terms *display* and *screen* are not equivalent. A display may consist of more than one screen. This feature might be implemented in several ways. There might be two physical monitors, linked to form a single display, as shown in Figure 1-7. Alternatively, two screens might be defined as different ways of using the same physical monitor. For example, on the Sun-3/60 color workstation, screen 0 is color, and screen 1 is black and white. Each screen is the size of the monitor; you can only view one screen at a time. In practice, the two screen seem to be side by side: you can "scroll" between them by moving the pointer off either horizontal edge of the screen. By default, windows are always placed on screen 0 but you can place a client window on screen 1 by specifying the screen number in the `-display` option when starting the client. (See Chapter 3 for instructions on using the `-display` option.)

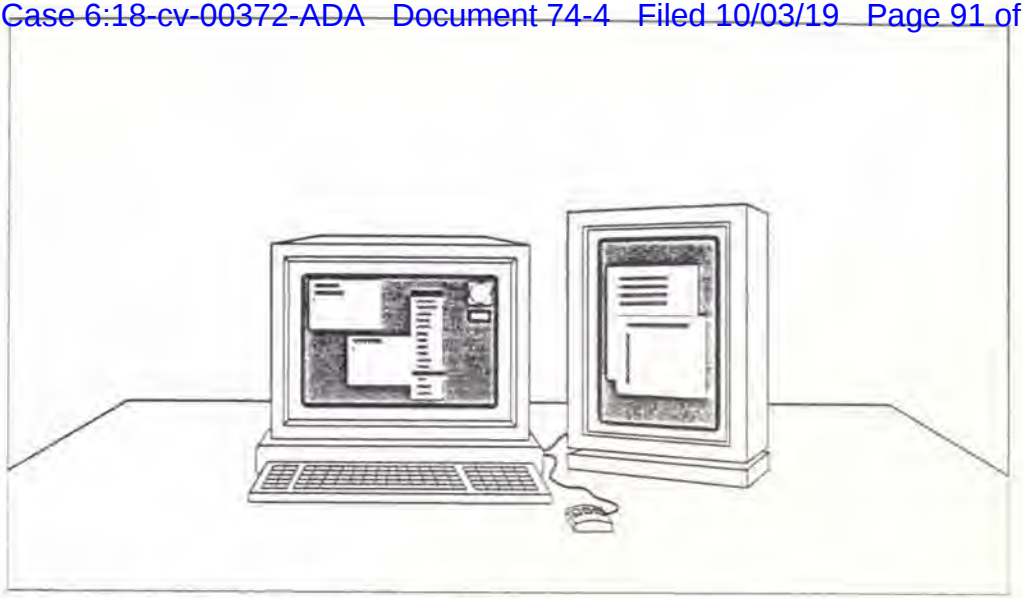


Figure 1-7. A display made up of two physical screens

Standard X Clients versus Motif Clients

The window manager running on a display helps determine the "look and feel" of an application. The *mwm* window manager frames each window on the display and allows you to manage a variety of application windows using the same mechanisms.

However, the look and feel of an application is not wholely determined by the window manager. In addition, the programming routines used to create the application also distinguish it. With the exception of *mwm*, all the applications we've looked at so far have been written (or rewritten) using what is known as the X Toolkit, developed at MIT.

The X Toolkit is a collective name for two subroutine libraries designed to simplify the development of X Window System applications: the X Toolkit (Xt) Intrinsics and the Athena widget set (Xaw). The Xt library consists of low-level C programming routines for building and using widgets, which are pre-defined user interface components or objects. Typical widgets create graphical features such as menus, command buttons, dialog boxes, and scrollbars. Widgets make it easier to create complex applications. A common widget set also ensures a consistent user interface between applications.

Remember that X does not provide a distinct graphical user interface (GUI). X is a basic window system upon which almost any style of GUI can be built. The X Toolkit provides a simplified approach to creating graphical user interface components—guidelines for writing and implementing widgets—rather than offering a set of components with a predefined look and feel. (However, the Athena widget set does provide X Toolkit applications with certain common features, many of which are mentioned in Chapter 8.)

In response to the need for a graphical user interface for X, the Open Software Foundation developed the Motif Toolkit. The Motif Toolkit is based upon the Xt Intrinsics and upon widget sets originally developed by two OSF sponsor companies, Digital Equipment Corporation and Hewlett-Packard. The Motif widget set was designed to emulate the look and feel of the IBM/Microsoft Presentation Manager, popular in the microcomputer world. An application coded using the Motif Toolkit has a distinct look and feel.

AT&T and Sun Microsystems have also developed a GUI—or more precisely, a specification for a GUI—called OPEN LOOK. At present the two major implementations of the OPEN LOOK specification are Sun's XView toolkit (which is not Xt based) and AT&T's OPEN LOOK widget set (which is Xt based). OPEN LOOK and Motif are the prime contenders to establish a graphical user interface standard in the market.

With the exception of *mwm* and a program called *mre* (the Motif resource editor), all of the clients discussed in this guide are standard X clients shipped by MIT. Most of these clients have been built with the X Toolkit and illustrate the use of many of the Athena widgets. When you run these standard clients with the *mwm* window manager, your environment is something of a hybrid—neither a vanilla X nor a Motif environment (with Motif applications in addition to the window manager).

A standard X client running with *mwm* is different from a true Motif application, one coded using the Motif Toolkit. At first look, they may seem very similar—when *mwm* is running all clients on the display are framed in the same way. In addition, certain graphical features provided by the Motif widgets are also provided, albeit with slight variations, by the Athena widgets. However, other features are unique to Motif.

Without dissecting every component or closely examining how it functions, let's briefly compare a standard X application to a Motif application, highlighting some of the major differences (primarily in appearance). Many features of Motif and standard X applications also *operate* differently. We'll examine the functionality of various Motif and Athena widgets in more detail in Chapter 8, *Other Clients*.

For the comparison, we'll use a Motif demo program called *mre*, the Motif resource editor. Developed at OSF by Mitch at OSF by Mitch Trachtenberg, *mre* assists you in editing your own resource specification file. The editing help *mre* provides is minimal, but the program clearly demonstrates many of the Motif widgets—as it was intended to do.

The standard X client we're using is *xclipboard*, which you use in concert with *xterm*'s "cut and paste" facility, described in Chapter 5. The *xclipboard* client provides a window in which you can paste multiple text selections and from which you can copy text selections to other windows. Similar to the clipboard feature of the Macintosh operating system, *xclipboard* is basically a storehouse for text you may want to paste into other windows, perhaps multiple times.

The *mre* window in Figure 1-8 contains a resource file to be edited. (Resources are a mechanism that allow you to customize the operation of X clients.) The *xclipboard* window in Figure 1-9 contains a long text selection "cut" from an *xterm* window. Some prominent features of each application are labelled. Both clients are illustrated without the *mwm* frame. (When the window manager is running, the frame creates a superficial resemblance among all clients on the display.)

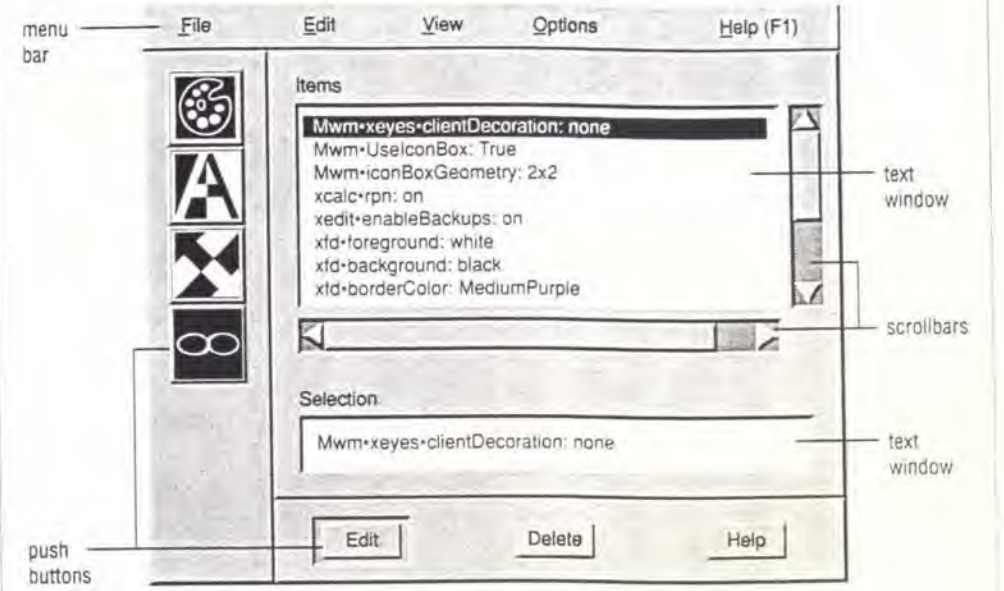


Figure 1-8. A Motif application: mre

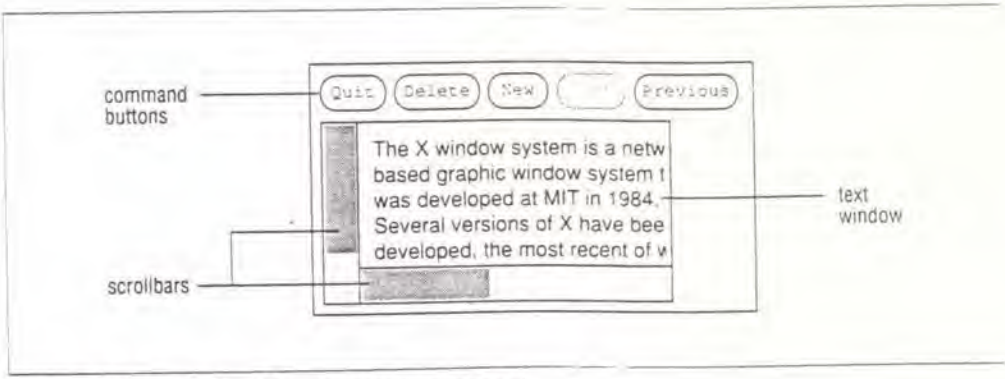


Figure 1-9. A standard X application: xclipboard

The most striking difference between the two clients is simply the amount of detail. Like the Motif window manager, the Motif widgets lend a three-dimensional appearance. The subwindows labelled Items and Selection seem to be set in to the application window. The push buttons (and drawn buttons) are shaded to suggest that they are raised above the level of the application window. (The drawn buttons also feature bitmapped images, three of them rather elaborate.) The menu bar is shaded to appear raised. The scrollbars have clearly distinguishable components, all of which are shaded and contoured to maintain the 3-D impression.

By contrast, the *xclipboard* window seems almost like a preliminary sketch of an application. It is basically flat. The text window, command buttons, and scrollbars are rendered in simple lines, without contouring, and with virtually no shading (though a portion of each scrollbar is shaded).

Another difference is that the *mre* window has a menu bar. Each word on the menu bar is the title of a pull-down menu that you can access by placing the pointer on it and clicking (or pressing and holding down) the first pointer button. The *xclipboard* application doesn't provide any menus—it's a fairly simple program. However, some standard X clients (notably *xman* discussed in Chapter 8) provide pull-down menus accessed by pressing and holding down a pointer button.

Motif pull-down menus generally have a few advantages over pull-down menus provided by standard X clients. While you must press and hold down a pointer button to dislay a menu provided by a standard X client, you can display a Motif menu simply by clicking a pointer button—and the menu stays displayed until you click again. Motif menu items can also be invoked in multiple ways (including pointer actions and keystrokes); the only way to invoke an item from a standard X menu is by dragging the pointer down the menu and releasing the button. The various ways you can work with a Motif pull-down menu are described for *mwm*'s Window Menu in Chapter 4.

Despite differences in general appearance and complexity, *mre* and *xclipboard* have many analogous components. Both applications feature a subwindow containing text that can be edited. (*mre* actually has two such windows, labeled Items and Selection.)

Both applications feature buttons: push buttons in the *mre* window; command buttons in the *xclipboard* window. From a user's viewpoint, push buttons and command buttons are functionally equivalent (though you can invoke a push button's function in more ways). The Motif and Athena widget sets simply identify them by different names. The four push buttons on the left side of the *mre* window are actually called drawn buttons. Drawn buttons are just push buttons decorated with a bitmap rather than a text label.

Both the *mre* and *xclipboard* windows have a horizontal and a vertical scrollbar, which are used to look at text that is currently outside the viewing window. (These scrollbars are only displayed when the text read into the window extends beyond the bounds of the viewing area. If the text only exceeds the viewing area in one direction—either horizontally or vertically—only one scrollbar will be displayed.) The Athena scrollbar is basically rectangular (actually one rectangle within another). The Motif scrollbar is somewhat more elaborate. Notice the arrows on either end, for instance. These arrows are the hallmark of a Motif scrollbar and can help you readily identify a Motif application. The arrows also provide functionality not duplicated by the Athena scrollbar.

In general, once you've mastered the basics of working with the MIT client programs running under the Motif Window Manager, you should have no problem making use of many additional features provided by commercial applications built with the Motif widget set.

X Architecture Overview

Most window systems are *kernel-based*: that is, they are closely tied to the operating system itself and can only run on a discrete system, such as a single workstation. The X Window System is not part of any operating system but instead is composed entirely of user-level programs.

The architecture of the X Window System is based on what is known as a *client-server* model. The system is divided into two distinct parts: *display servers* that provide display capabilities and keep track of user input and *clients*, application programs that perform specific tasks.

In a sense, the server acts as intermediary between client application programs, and the local display hardware (one or perhaps multiple screens) and input devices (generally a keyboard and pointer). When you enter input using the keyboard or a pointing device, the server conveys the input to the relevant client application. Likewise, the client programs make requests (for information, processes, etc.) that are communicated to the hardware display by the server. For example, a client may request that a window be moved or that text be displayed in the window.

This division within the X architecture allows the clients and the display server either to work together on the same machine or to reside on different machines (possibly of different types, with different operating systems, etc.) that are connected by a network. For example, you might use a relatively low-powered PC or workstation as a display server to interact with clients that are running on a more powerful remote system. Even though the client program is actually running on the more powerful system, all user input and displayed output occur on the PC or workstation server and are communicated across the network using the X protocol. Figure 1-10 shows a diagram of such a network.

You might choose to run a client on a remote machine for any number of reasons. Generally, however, the remote machine offers some feature unavailable on your local machine: a more efficient or powerful processor; a completely different architecture better suited to a particular task; different application software; file server capabilities (and perhaps large data files you'd rather not transfer over the network). X allows you to take advantage of these remote features and to see the results on your local terminal.

The distinction between clients and the server also allows somewhat complicated display situations. For instance, you can access several machines simultaneously. (This can greatly simplify the work of a system administrator.) X also allows you to output to several displays simultaneously. This capability can be very helpful in educational situations. Hypothetically, a teacher could display instructional material to a group of students each using a graphics workstation or terminal hooked up to a network.

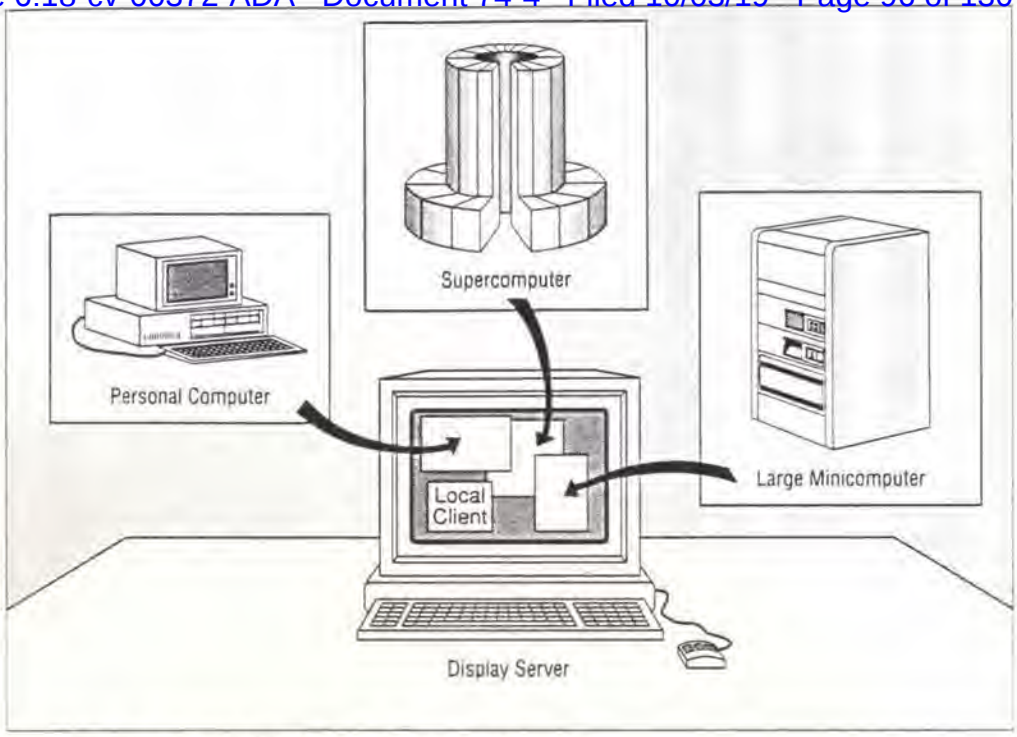


Figure 1-10. A sample X Window System configuration

There is another less obvious advantage to the client-server model: since the server is entirely responsible for interacting with the hardware, only the server program must be machine-specific. X client applications can easily be ported from system to system.

The X Display Server

The X display server is a program that keeps track of all input from input devices, such as the keyboard and pointer, and conveys that input to the relevant client applications; the server also keeps track of output from any clients that are running and updates the display to reflect that output. Each physical display (which may be multiple screens) has only one server program.

User input and several other types of information pass from the server to a client in the form of *events*. An event is a packet of information that tells the client something it needs to act on, such as keyboard input. Moving the pointer or pressing a key, etc., causes *input* events to occur.

When a client program receives a meaningful event, it responds with requests to the server for some sort of action affecting the display. For instance, the client may request that a window be resized to particular dimensions. The server responds to requests by a client program by updating the appropriate window(s) on your display.

Servers are available for PCs, workstations, and even for special terminals (generally called X terminals), which may have the server downloaded from another machine or stored in ROM.

Clients

As previously mentioned, a client is an application program. The standard release of X from MIT includes more than 50 client programs that perform a wide variety of tasks. X allows you to run many clients simultaneously: each client displays in a separate window. For example, you could be editing a text file in one window, compiling a program source file in a second window, reading your mail in a third, all the while displaying the system load average in a fourth window.

While X clients generally display their results and take input from a single display server, they may each be running on a different computer on the network. It is important to note that the same programs may not look and act the same on different servers since X has no standard user interface, since users can customize X clients differently on each server, and since the display hardware on each server may be different.

Remember that the server conveys input from the various input devices to the appropriate client application; likewise, the client issues output in the form of requests to the server for certain actions affecting the display.

In addition to communicating with the server, a client sometimes need to communicate with other clients. For example, a client may need to tell the window manager where to place its icon. Interclient communication is facilitated by the use of *properties*. A property is a piece of information associated with a window or a font and stored in the server. Properties are used by clients to store information that other clients might need to know, such as the name of the application associated with a particular window. Storing properties in the server makes the information they contain accessible to all clients.

A typical use of properties in interclient communication involves how a client tells the window manager the name of the application associated with its window. By default, the application name corresponds to the client's name, but many clients allow you to specify an alternative name when you run the program. A window manager that provides a titlebar needs to know the application name to display in that area. The client's application name is stored in the server in the property called WM_NAME and is accessed by the window manager.

See the *xprop* reference page in Part Three of this guide, and Volume One, *Xlib Programming Manual*, for more information about properties and the *xprop* client.

Several of the more frequently used client programs are discussed in the following sections.

The Window Manager

The way a kernel-based window system operates is inherent in the window system itself. By contrast, the X Window System concentrates control in a special kind of client application, called a window manager. The window manager you use largely determines the look and feel of X on a particular system.

The window manager shipped with OSF/Motif is called *mwm*. As we've discussed, *mwm* allows you to move and resize windows, rearrange the order of windows in the window stack, create additional windows, and convert windows into icons, etc. These functions are discussed more fully in Chapter 3 and Chapter 4.

mwm is compliant with the X Consortium's *Inter-Client Communication Conventions Manual* (ICCCM), introduced at Release 3. The ICCCM contains standards for interaction with window managers and other clients. It defines basic policy intentionally omitted from X itself, such as the rules for transferring data between applications, for transferring keyboard focus, for installing colormaps, and so on. As long as applications and window managers follow the conventions outlined in the ICCCM, applications created with different toolkits will be able to coexist and work together on the same server.

In this guide, we assume you are using *mwm*. Other popular window managers, such as *twm* (the tab window manager), *awm* (Ardent™ window manager), *rtl* (tiled window manager, developed at Siemens Research and Technology Laboratories, RTL), and *olwm* (the OPEN LOOK™ window manager), are also widely used.

If the *mwm* window manager has been customized at your site or you are using a different window manager, many of the concepts should remain the same. However, the actual procedures shown may differ. See Chapter 11, *Customizing mwm*, for a discussion of how to tailor *mwm* to your particular needs.

The xterm Terminal Emulator

X11 itself is designed to support only bitmapped graphics displays. For this reason, one of the most important clients is a terminal emulator. The terminal emulator brings up a window that allows you to log in to a multiuser system and to run applications designed for use on a standard alphanumeric terminal. Anything you can do on a terminal, you can do in this window.

xterm is the most widely available terminal emulator. *xterm* emulates a DEC® VT102 terminal or a Tektronix® 4014 terminal. For each *xterm* process, you can display both types of windows at the same time but only one is active (i.e., responding to input) at a time.

Running multiple *xterm* processes is like working with multiple terminals. Since you can bring up more than one *xterm* window at a time, you can run several programs simultaneously. For example, you can have the system transfer files or process information in one window while you focus your attention on a text-editing session in another window. As you might imagine, having what are in effect multiple terminals can increase your productivity remarkably. See Chapter 3, *Working in the X Environment*, and Chapter 5, *The xterm Terminal Emulator*, for more information about *xterm*.

The display manager, *xdm*, is a client that is designed to start the X server automatically (from the UNIX */etc/rc* system startup file) and to keep it running. (X can also be started manually, as described in Chapter 2, *Getting Started*.) In its most basic implementation, the display manager emulates the *getty* and *login* programs, which put up the login prompt on a standard terminal, keeping the server running, prompting for a user's name and password, and managing a standard login session.

However, *xdm* has far more powerful and versatile capabilities. Users can design their own sessions, automatically running several clients and setting personal resources (such as keyboard, pointer, and display characteristics). The system administrator can also customize special *xdm* files to manage several connected displays (both local and remote) and to set system-wide X resources (for example, client default features). Resources are introduced in Chapter 3, *Working in the X Environment*, and discussed fully in Chapter 10, *Setting Resources*.

Many commercial vendors provide alternative display/session managers. If you are using a display manager other than *xdm*, many of the concepts should remain the same. However, the actual setup procedures may differ. See Appendix A, *System Management*, for a discussion of how to set up and customize the *xdm* display manager.

Other X Clients

The standard distribution of X from MIT includes more than 50 client applications. The client you will probably used most frequently is *xterm*. We've grouped some of the other more useful applications as follows:

Desk accessories

xbiff	Mail notification program.
xclock, oclock	Clock applications.
xcalc	Desktop calculator.
xload	System load monitor.
xman	Manual page browser.

Display and keyboard preferences

xset	Allows you to set various display and keyboard preferences, such as bell volume, cursor acceleration, and screen saver operation.
xmodmap	Allows you to map keyboard keys and pointer buttons to particular functions.

Font utilities

xlsfonts	Lists available fonts.
xfd	Displays the characters in a single font.
xfontsel	Allows you to display multiple fonts sequentially and select a font to be used by another application.

Graphics utilities

bitmap	Bitmap editor.
atobm, *bmtoa*	Programs to convert ASCII characters to bitmaps and bitmaps to ASCII characters.

Printing applications

xwd	Dumps the image of a window to a file.
xpr	Translates an image file produced by *xwd* to PostScript® or another format, suitable for printing on a variety of printers.
xwud	Redisplays a window dump file created using *xwd*.

Removing a window

xkill	Terminates a client application.

Window and display information utilities

xlsclients	Lists the clients running on the display.
xdpyinfo	Lists general characteristics of the display.
xwininfo	Lists general characteristics of a selected window.
xprop	Lists the properties associated with a window.

These and other client applications are described in Chapters 5 through 8, and in Chapter 11. In addition, a reference page describing each client and listing its options appears in Part Three of this guide. As more commercial and user-contributed software is developed, many more specialized programs will become available.

Customizing Clients

Most X clients are designed to be customized by the user. A multitude of command-line options can be used to affect the appearance and operation of a single client process. A few of the more useful command-line options are introduced in Chapter 3. Chapter 9 discusses several options in detail. Part Three of this guide includes a reference page for each client that details all valid options.

X also provides a somewhat more convenient way to customize the appearance and operation of client programs. Rather than specifying all characteristics using command line options, default values for most options can be stored in a file (generally called *.Xresources* or *.Xdefaults*) in your home directory. Each default value is set using a variable called a *resource*:

you can change the behavior or appearance of a program by changing the *value* associated with a resource variable.

Generally, these resource values are loaded into the server using a program called *xrdb* (*X* resource *d*ata*b*ase manager). Then the values are accessed automatically when you run a client. Storing your preferences in the server with *xrdb* also allows you to run clients on multiple machines without maintaining an *Xresources* file on each machine.

There is a separate customization file for the *mwm* window manager called *.mwmrc*, which is also kept in your home directory. By editing the *.mwmrc* file, you can modify several aspects of the window manager's operation, such as the contents of menus, and the key and pointer button sequences used to invoke actions. See Chapter 11, *Customizing mwm*, for more information.

Client customization is introduced in Chapter 3, *Working in the X Environment*, and described fully in Part Two of this guide.

EXHIBIT J

Making Everything Easier!

Droid™ X

FOR DUMMIES®

Learn to:

- **Set up and configure your Droid X and synchronize it with your PC**
- **Call, set up contacts, text, e-mail, or do some social networking**
- **Listen to your favorite tunes or take amazing photos and videos**

IN FULL COLOR!

Dan Gookin
Bestselling author of PCs For Dummies
and Laptops For Dummies

Get More and Do More at Dummies.com®

Start with **FREE** Cheat Sheets

Cheat Sheets include

- Checklists
- Charts
- Common Instructions
- And Other Good Stuff!

To access the Cheat Sheet created specifically for this book, go to
www.dummies.com/cheatsheet/droidx

Get Smart at Dummies.com

Dummies.com makes your life easier with 1,000s of answers on everything from removing wallpaper to using the latest version of Windows.

Check out our

- Videos
- Illustrated Articles
- Step-by-Step Instructions

Plus, each month you can win valuable prizes by entering our Dummies.com sweepstakes.*

Want a weekly dose of Dummies? Sign up for Newsletters on

- Digital Photography
- Microsoft Windows & Office
- Personal Finance & Investing
- Health & Wellness
- Computing, iPods & Cell Phones
- eBay
- Internet
- Food, Home & Garden

Find out "HOW" at Dummies.com

**Sweepstakes not currently available in all countries; visit Dummies.com for official rules.*

Droid™ X

FOR

DUMMIES®

by Dan Gookin

Wiley Publishing, Inc.

Droid™ X For Dummies®

Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Control Number: 2010932456

ISBN: 978-0-470-90319-3

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

There's No Screen Like Home

The first thing you see after you unlock your Droid X is the *Home* screen, illustrated in Figure 3-2. The Home screen is also the location you go to whenever you end a call or quit an application.

Figure 3-2: The Home screen.

Here are the key items to notice on the Home screen (refer to Figure 3-2):

Status bar: The top of the Home screen is a thin, informative strip I call the *status bar*. It contains notification icons and status icons, plus the current time.

Notification icons: These icons come and go, depending on what happens in your digital life. For example, a new icon appears whenever you receive a new email message or have a pending appointment. The section "Reviewing notifications," later in this chapter, describes how to deal with notifications.

Status icons: These icons represent the phone's current condition, such as the type of network it's connected to, its signal strength, and its battery status, as well as whether the speaker has been muted or a Wi-Fi network is connected.

Widgets: A *widget* is a teensy program that can display information, let you control the phone, access features, or do something purely amusing. You can read more about widgets in Chapter 21.

Application icons: The meat of the meal on the Home screen plate is the collection of application icons. Touching an icon runs the program.

Launcher: Touching the Launcher button icon displays the Applications Tray, a scrolling list of all applications installed on your phone. The section "The Applications Tray," later in this chapter, describes how it works.

And now, the secret: The Home screen is seven times wider than what you see on the front of your Droid X. It has left and right wings to the Home screen, as illustrated in Figure 3-3.

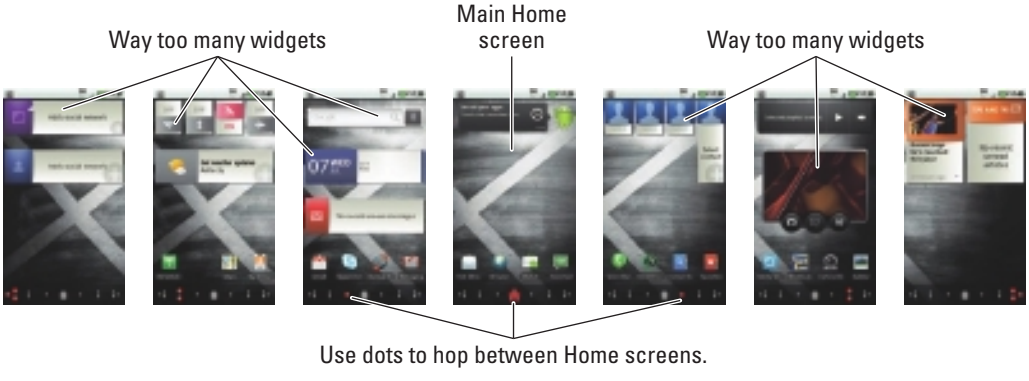


Figure 3-3: All the Home screens.

To view the left and right sides of the Home screen, swipe your finger left or right across the touchscreen display. The Home screen slides over one page in each direction every time you swipe.

The wider Home screen gives you more opportunities to place applications and widgets. As shown in Figure 3-3, the Droid X Home screens hold a lot of widgets. Refer to Chapter 21 for information about deleting widgets you don't need and for other information about customizing the Home screen.

Touching part of the Home screen that doesn't feature an icon or a control doesn't do anything — unless you're using the *Live Wallpaper* feature. In that case, touching the screen changes the wallpaper in some

way, depending on the selected wallpaper. You can read more about Live Wallpaper in Chapter 21.

- The variety of notification and status icons is broad. You see the icons referenced in appropriate sections throughout this book.
- No matter which part of the Home screen you're viewing, the top part of the touchscreen stays the same — the status bar always displays notification and status icons and the time.
- To return to the Home screen at any time, press the Home soft button.

I've Been Working on the Home Screen

I recommend getting to know three basic Home screen operations: reviewing notifications, starting applications, and accessing widgets.

Reviewing notifications

Notifications appear as icons at the top of the Home screen, as illustrated earlier, in Figure 3-2. To see the actual notifications, peel down the top part of the screen, as shown in Figure 3-4.

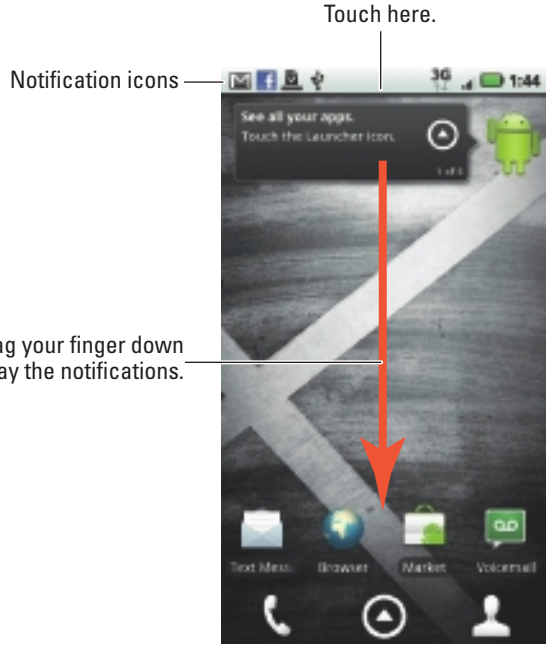


Figure 3-4: Accessing notifications.

EXHIBIT K

answer mode *n.* A setting that allows a modem to answer an incoming call automatically. It is used in all fax machines. *Also called:* auto answer.

answer-only modem *n.* A modem that can receive but not originate calls.

answer/originate modem *n.* A modem that can both send and receive calls—the most common type of modem in use.

antialiasing *n.* A software technique for smoothing the jagged appearance of curved or diagonal lines caused by poor resolution on a display screen. Methods of anti-aliasing include surrounding pixels with intermediate shades and manipulating the size and horizontal alignment of pixels. See the illustration. *See also* dithering. *Compare* aliasing.

Antialiasing. *The image on the right shows the result of anti-aliasing through the use of higher resolution.*

antiglare or **anti-glare** *adj.* Pertaining to any measure taken to reduce reflections of external light on a monitor screen. The screen may be coated with a chemical (which may reduce its brightness), covered with a polarizing filter, or simply rotated so that external light is not reflected into the user's eye.

anti-replay *n.* An IP packet–level security feature that prevents packets that have been intercepted and changed from being inserted into the data stream. Anti-replay creates a security association between a source and destination computer, with each agreeing on a numbering sequence for transmitted packets. The anti-replay mechanism detects packets tagged with numbers that fall outside the accepted sequence, discards them, sends an error message, and logs the event. The anti-replay protocol is included as part of the IPSec standard. *See also* IPSec.

antistatic device *n.* A device designed to minimize shocks caused by the buildup of static electricity, which can disrupt computer equipment or cause data loss. An antistatic device may take the form of a floor mat, a wristband with a wire attached to the workstation, a spray, a lotion, or other special-purpose device. *See also* static[2], static electricity.

antivirus program *n.* A computer program that scans a computer's memory and mass storage to identify, isolate, and eliminate viruses, and that examines incoming files for viruses as the computer receives them.

anti-worm *n. See* automatic patching, do-gooder virus.

anycasting *n.* Communication between a single sender and the nearest receiver in a group. In IPv6, anycasting enables one host to initiate the updating of routing tables for a group of hosts. *See also* IPv6. *Compare* multicasting, unicast.

any key *n.* Any random key on a computer keyboard. Some programs prompt the user to "press any key" to continue. It does not matter which key the user presses. There is no key on the keyboard called Any.

any-to-any connectivity *n.* The property of an integrated computer network environment where it is possible to share data across multiple protocols, host types, and network topologies.

AOL *n. See* America Online.

AOL Instant Messenger *n. See* AIM.

AOL NetFind *n.* Resident Web-finding tool of America Online (AOL) information service. Searches by keyword and concept. Using Intelligent Concept Extraction (ICE) and Excite technology, this tool finds relationships between words and ideas; for example, between "elderly people" and "senior citizen." *See also* Excite, Intelligent Concept Extraction.

APA *n. See* all points addressable.

Apache *n.* A free open-source HTTP (Web) server introduced in 1995 by the Apache Group as an extension to, and improvement of, the National Center for Supercomputing Applications' earlier HTTPd (version 1.3). Apache is popular on UNIX-based systems, including Linux, and also runs on Windows NT and other operating systems, such as BeOS. Because the server was based on existing code with a series of patches, it became known as "A Patchy server," which led to the official name Apache. *See also* HTTPd.

EXHIBIT L

Vital Information for Apache Programmers & Administrators

Apache

The Definitive Guide

O'REILLY

Ben Laurie & Peter Laurie

Apache: The Definitive Guide

Ben Laurie & Peter Laurie

Second Edition, February 1999, updated February 2000

ISBN: 1-56592-528-9, 388 pages

Written and reviewed by key members of the Apache group, this book is the only complete guide on the market that describes how to obtain, set up, and secure the Apache software on both Unix and Windows systems.

The second edition fully describes Windows support and all the other Apache 1.3 features.

The freeware Apache web server runs on about half of the world's existing web sites, and it is rapidly increasing in popularity. *Apache: The Definitive Guide*, written and reviewed by key members of the Apache Group, is the only complete guide on the market today that describes how to obtain, set up, and secure the Apache software.

Apache was originally based on code and ideas found in the most popular HTTP server of the time: NCSA httpd 1.3 (early 1995). It has since evolved into a far superior system that can rival (and probably surpass) almost any other UNIX-based HTTP server in terms of functionality, efficiency, and speed. The new version now includes support for Win32 systems. This new second edition of *Apache: The Definitive Guide* fully describes Windows support and all the other Apache 1.3 features. Contents include:

- The history of the Apache Group
- Obtaining and compiling the server
- Configuring and running Apache on UNIX and Windows, including such topics as directory structures, virtual hosts, and CGI programming
- The Apache 1.3 Module API
- Apache security
- A complete list of configuration directives
- A complete demo of a sample web site

With *Apache: The Definitive Guide*, web administrators new to Apache can get up to speed more quickly than ever before by working through the tutorial demo. Experienced administrators and CGI programmers, and web administrators moving from UNIX to Windows, will find the reference sections indispensable. *Apache: The Definitive Guide* is the definitive documentation for the world's most popular web server.

Preface

Apache: The Definitive Guide is principally about the Apache web server software. We explain what a web server is and how it works, but our assumption is that most of our readers have used the World Wide Web and understand in practical terms how it works, and that they are now thinking about running their own servers to offer material to the hungry masses.

This book takes the reader through the process of acquiring, compiling, installing, configuring, and modifying Apache. We exercise most of the package's functions by showing a set of example sites that take a reasonably typical web business - in our case, a postcard publisher - through a process of development and increasing complexity. However, we have deliberately not tried to make each site more complicated than the last. Most of the chapters refer to an illustrative site that is as simple as we could make it. Each site is pretty well self-contained so that the reader can refer to it while following the text without having to disentangle the meat there from extraneous vegetables. If desired, it is perfectly possible to install and run each site on a suitable system.

Perhaps it is worth saying what this book is *not*. It is not a manual, in the sense of formally documenting every command - such a manual exists on the Apache site and has been much improved with Version 1.3; we assume that if you want to use Apache, you will download it and keep it at hand. Rather, if the manual is a roadmap that tells you how to get somewhere, this book tries to be a tourist guide that tells you why you might want to make the journey.

It also is *not* a book about HTML or creating web pages, or one about web security or even about running a web site. These are all complex subjects that should either be treated thoroughly or left alone. A compact, readable book that dealt *thoroughly* with all these topics would be most desirable.

A webmaster's library, however, is likely to be much bigger. It might include books on the following topics:

- The Web and how it works
- HTML - what you can do with it
- How to decide what sort of web site you want, how to organize it, and how to protect it
- How to implement the site you want using one of the available servers (for instance, Apache)
- Handbooks on Java, Perl, and other languages
- Security

Apache: The Definitive Guide is just one of the six or so possible titles in the fourth category.

Apache is a versatile package and is becoming more versatile every day, so we have not tried to illustrate every possible combination of commands; that would require a book of a million pages or so. Rather, we have tried to suggest lines of development that a typical webmaster should be able to follow once an understanding of the basic concepts is achieved.

As with the first edition, writing the book was something of a race with Apache's developers. We wanted to be ready as soon as Version 1.3 was stable, but not before the developers had finished adding new features. Unfortunately, although 1.3 was in "feature freeze" from early 1998 on, we could not be sure that new features might not become necessary to fix newly discovered problems.

In many of the examples that follow, the motivation for what we make Apache do is simple enough and requires little explanation (for example, the different index formats in Chapter 7). Elsewhere, we feel that the webmaster needs to be aware of wider issues (for instance, the security issues discussed in Chapter 13) before making sensible decisions about his or her site's configuration, and we have not hesitated to branch out to deal with them.

Who Wrote Apache, and Why?

Apache gets its name from the fact that it consists of some existing code plus some *patches*. The FAQ[1] thinks that this is cute; others may think it's the sort of joke that gets programmers a bad name. A more responsible group thinks that Apache is an appropriate title because of the resourcefulness and adaptability of the American Indian tribe.

You have to understand that Apache is free to its users and is written by a team of volunteers who do not get paid for their work. Whether or not they decide to incorporate your or anyone else's ideas is entirely up to them. If you don't like this, feel free to collect a team and write your own web server.

The first web server was built by the British physicist Tim Berners-Lee at CERN, the European Centre for Nuclear Research at Geneva, Switzerland. The immediate ancestor of Apache was built by the U.S. government in the person of NCSA, the National Center for Supercomputing Applications. This fine body is not to be confused with the National Computing Security Agency or the North Carolina Schools Association. Because this code was written with (American) taxpayers' money, it is available to all; you can, if you like, download the source code in C from www.ncsa.uiuc.edu, paying due attention to the license conditions.

There were those who thought that things could be done better, and in the FAQ for Apache (at http://www.apache.org) we read:

...Apache was originally based on code and ideas found in the most popular HTTP server of the time, NCSA httpd 1.3 (early 1995).

That phrase "of the time" is nice. It usually refers to good times back in the 1700s or the early days of technology in the 1900s. But here it means back in the deliquescent bogs of a few years ago!

While the Apache site is open to all, Apache is written by an invited group of (we hope) reasonably good programmers. One of the authors of this book, Ben, is a member of this group.

Why do they bother? Why do these programmers, who presumably could be well paid for doing something else, sit up nights to work on Apache for our benefit? There is no such thing as a free lunch, so they do it for a number of typically human reasons. One might list, in no particular order:

- They want to do something more interesting than their day job, which might be writing stock control packages for BigBins, Inc.
- They want to be involved on the edge of what is happening. Working on a project like this is a pretty good way to keep up-to-date. After that comes consultancy on the next hot project.
- The more worldly ones might remember how, back in the old days of 1995, quite a lot of the people working on the web server at NCSA left for a thing called Netscape and became, in the passage of the age, zillionaires.
- It's fun. Developing good software is interesting and amusing and you get to meet and work with other clever people.
- They are not doing the bit that programmers hate: explaining to end users why their treasure isn't working and trying to fix it in 10 minutes flat. If you want support on Apache you have to consult one of several commercial organizations (see Appendix A), who, quite properly, want to be paid for doing the work everyone loathes.

[1] *FAQ is netspeak for Frequently Asked Questions. Most sites/subjects have an FAQ file that tells you what the thing is, why it is, and where it is going. It is perfectly reasonable for the newcomer to ask for the FAQ to look up anything new to him or her, and indeed this is a sensible thing to do, since it reduces the number of questions asked. Apache's FAQ can be found at http://www.apache.org/docs/FAQ.html.*

Conventions Used in This Book

This section covers the various conventions used in this book.

Typographic Conventions

`Constant Width`

Used for HTTP headers, status codes, MIME content types, directives in configuration files, commands, options/switches, functions, methods, variable names, and code within body text

`Constant Width Bold`

Used in code segments to indicate input to be typed in by the user

`Constant Width Italic`

Used for replaceable items in code and text

Italic

Used for filenames, pathnames, newsgroup names, Internet addresses (URLs), email addresses, variable names (except in examples), terms being introduced, program names, subroutine names, CGI script names, hostnames, usernames, and group names

Icons

Text marked with this icon applies to the Unix version of Apache.

Text marked with this icon applies to the Win32 version of Apache.


The owl symbol designates a note relating to the surrounding text.


The turkey symbol designates a warning related to the surrounding text.

Pathnames

We use the text convention *... /* to indicate your path to the demonstration sites, which may well be different from ours. For instance, on our Apache machine, we kept all the demonstration sites in the directory */usr/www.* So, for example, our path would be */usr/www/site.simple*. You might want to keep the sites somewhere other than */usr/www,* so we refer to the path as *... /site.simple.*

Don't type .../ into your computer. The attempt will upset it!

Directives

Apache is controlled through roughly 150 directives. For each directive, a formal explanation is given in the following format:

Directive

```
Syntax
Where used
```

An explanation of the directive is located here.

So, for instance, we have the following directive:

ServerAdmin

```
ServerAdmin email address
Server config, virtual host
```

`ServerAdmin` gives the email address for correspondence. It automatically generates error messages so the user has someone to write to in case of problems.

The "where used" line explains the appropriate environment for the directive. This will become clearer later.

Organization of This Book

The chapters that follow and their contents are listed here:

Chapter 1

Covers web servers, how Apache works, TCP/IP, HTTP, hostnames, what a client does, what happens at the server end, choosing a Unix version, and compiling and installing Apache under both Unix and Win32.

Chapter 2

Discusses getting Apache to run, creating Apache users, runtime flags, permissions, and *site.simple*.

Chapter 3

Introduces a demonstration business, Butterthlies, Inc.; some HTML; default indexing of web pages; server housekeeping; and block directives.

Chapter 4

Demonstrates aliases, logs, HTML forms, shell script, a CGI in C, environment variables, and adapting to the client's browser.

Chapter 5

Explains controlling access, collecting information about clients, cookies, DBM control, digest authentication, and anonymous access.

Chapter 6

Covers content and language arbitration, type maps, and expiration of information.

Chapter 7

Discusses better indexes, index options, your own indexes, and imagemaps.

Chapter 8

Describes `Alias`, `ScriptAlias`, and the amazing `Rewrite` module.

Chapter 9

Covers remote proxies and proxy caching.

Chapter 10

Explains runtime commands in your HTML and XSSI - a more secure server-side include.

Chapter 11

Covers server status, logging the action, and configuring the log files.

Chapter 12

Discusses authentication, blocking, counters, faster CGI, languages, server-side scripting, and URL rewriting.

Chapter 13

Discusses Apache's security precautions, validating users, binary signatures, virtual cash, certificates, firewalls, packet filtering, secure sockets layer (SSL), legal issues, patent rights, national security, and Apache-SSL directives.

Chapter 14

Describes pools; per-server, per-directory, and per-request information; functions; warnings; and parsing.

Chapter 15

Covers status codes; module structure; the command table; the initializer, translate name, check access, check user ID, check authorization and check type routines; prerun fixups; handlers; the logger; and a complete example.

Appendix A

Provides a list of commercial service and/or consultation providers.

Appendix B

Provides a listing of *echo.c*.

Appendix C

Contains Apache Group internal mail discussing NCSA/Apache compatibility issues.

Appendix D

Provides the SSL specification.

Appendix E

Contains a listing of the full log file referenced in Chapter 11.

In addition, the Apache Quick Reference Card provides an outline of the Apache 1.3.4 syntax.

Acknowledgments

First, thanks to Robert S. Thau, who gave the world the Apache API and the code that implements it, and to the Apache Group, who worked on it before and have worked on it since. Thanks to Eric Young and Tim Hudson for giving SSLeay to the Web.

Thanks to Bryan Blank, Aram Mirzadeh, Chuck Murcko, and Randy Terbush, who read early drafts of the first edition text and made many useful suggestions; and to John Ackermann, Geoff Meek, and Shane Owenby, who did the same for the second edition. Thanks to Paul C. Kocher for allowing us to reproduce SSL Protocol, Version 3.0, in Appendix D, and to Netscape Corporation for allowing us to reproduce *echo.c* in Appendix B.

We would also like to offer special thanks to Andrew Ford for giving us permission to reprint his Apache Quick Reference Card.

Many thanks to Robert Denn, our editor at O'Reilly, who patiently turned our text into a book - again. The two layers of blunders that remain are our own contribution.

And finally, thanks to Camilla von Massenbach and Barbara Laurie, who have continued to put up with us while we rewrote this book.

Chapter 9. Proxy Server

An important concern on the Web is keeping the Bad Guys out of your network (see Chapter 13). One established technique is to keep the network hidden behind a firewall; this works well, but as soon as you do it, it also means that everyone on the same network suddenly finds that their view of the Net has disappeared (rather like people living near Miami Beach before and after the building boom). This becomes an urgent issue at Buttherthlies, Inc., as competition heats up and naughty-minded Bad Guys keep trying to break our security and get in. We install a firewall and, anticipating the instant outcries from the marketing animals who need to get out on the Web and surf for prey, we also install a proxy server to get them out there.

So, in addition to the Apache that serves clients visiting our sites and is protected by the firewall, we need a copy of Apache to act as a proxy server to let us, in our turn, access other sites out on the Web. Without the proxy server, those inside are safe but blind.

9.1 Proxy Directives

We are not concerned here with firewalls, so we take them for granted. The interesting thing is how we configure the proxy Apache to make life with a firewall tolerable to those behind it.

site.proxy has three subdirectories: *cache, proxy, real.* The Config file from *... /site. proxy/proxy* is as follows:

```
User webuser
Group webgroup
ServerName www.butterthlies.com

Port 8000
ProxyRequests on
CacheRoot /usr/www/site.proxy/cache
CacheSize 100000
```

The points to notice are that:

- On this site we use `ServerName` *www.butterthlies.com.*
- The `Port` number is set to 8000 so that we can change proxies without having to change users' Configs.
- We turn `ProxyRequests` on and provide a directory for the cache, which we will discuss later in this chapter.
- `CacheRoot` is set up in a special directory.
- `CacheSize` is set to 100000 kilobytes.

9.1.1 ProxyRequests

```
ProxyRequests [on|off]
Default: off
Server config
```

This directive turns proxy serving on. Even if `ProxyRequests` is `off`, `ProxyPass` directives are still honored.

9.1.2 ProxyRemote

```
ProxyRemote remote-server = protocol://hostname[:port]
Server config
```

This directive defines remote proxies to this proxy. *remote-server* is either the name of a URL scheme that the remote server supports, a partial URL for which the remote server should be used, or " * " to indicate that the server should be contacted for all requests. *protocol* is the protocol that should be used to communicate with the remote server. Currently, only HTTP is supported by this module. For example:

```
ProxyRemote ftp http://ftpproxy.mydomain.com:8080
  ProxyRemote http://goodguys.com/ http://mirrorguys.com:8000
  ProxyRemote * http://cleversite.com
```

9.1.3 ProxyPass

```
ProxyPass path url
Server config
```

This command runs on an ordinary server and translates requests for a named directory and below to a demand to a proxy server. So, on our ordinary Butterthlies site, we might want to pass requests to */secrets* onto a proxy server *darkstar.com*:

```
ProxyPass /secrets http://darkstar.com
```

Unfortunately, this is less useful than it might appear, since the proxy does not modify the HTML returned by *darkstar.com*. This means that URLs embedded in the HTML will refer to documents on the main server unless they have been written carefully. For example, suppose a document *one.html* is stored on *darkstar.com* with the URL *http://darkstar.com/one.html*, and we want it to refer to another document in the same directory. Then the following links will work, when accessed as *http://www.butterthlies.com/secrets/one.html*:

```
<A HREF="two.html">Two</A>
<A HREF="/secrets/two.html">Two</A>
<A HREF="http://darkstar.com/two.html">Two</A>
```

But this example will not work:

```
<A HREF="/two.html">Not two</A>
```

When accessed directly, through *http://darkstar.com/one.html*, these links work:

```
<A HREF="two.html">Two</A>
<A HREF="/two.html">Two</A>
<A HREF="http://darkstar.com/two.html">Two</A>
```

But the following doesn't:

```
<A HREF="/secrets/two.html">Two</A>
```

9.1.4 ProxyDomain

```
ProxyDomain Domain
Server config
```

This directive is only useful for Apache proxy servers within intranets. The `ProxyDomain` directive specifies the default domain to which the Apache proxy server will belong. If a request to a host without a domain name is encountered, a redirection response to the same host with the configured *Domain* appended will be generated.

9.1.5 NoProxy

```
NoProxy { Domain | SubNet | IpAddr | Hostname }
Server config
```

This directive is only useful for Apache proxy servers within intranets. The `NoProxy` directive specifies a list of subnets, IP addresses, hosts, and/or domains, separated by spaces. A request to a host that matches one or more of these is always served directly, without forwarding to the configured `ProxyRemote` proxy server(s).

9.1.6 ProxyPassReverse

```
ProxyPassReverse path url
Server config, virtual host
```

A reverse proxy is a way to share load between several servers - the frontend server simply accepts requests and forwards them to one of several backend servers. The optional module *mod_rewrite* has some special stuff in it to support this. This directive lets Apache adjust the URL in the `Location` response header. If a `ProxyPass` (or *mod_rewrite*) has been used to do reverse proxying, then this directive will rewrite `Location` headers coming back from the reverse proxied server so that they look as if they came from somewhere else (normally this server, of course).

9.2 Caching

Another reason for using a proxy server is to cache data from the Web to save the bandwidth of the world's sadly overloaded telephone systems and therefore to improve access time on our server.

The directive `CacheRoot`, cunningly inserted in the Config file shown earlier, and the provision of a properly permissioned cache directory allow us to show this happening. We start by providing the directory *.../site.proxy/cache,* and Apache then improves on it with some sort of directory structure like *.../site.proxy/cache/d/o/j/gfqbZ@49rZiy6LOCw.*

The file *gfqbZ@49rZiy6LOCw* contains the following:

```
320994B6 32098D95 3209956C 00000000 0000001E
X-URL: http://192.168.124.1/message
HTTP/1.0 200 OK
Date: Thu, 08 Aug 1996 07:18:14 GMT
Server: Apache/1.1.1
Content-length: 30
Last-modified Thu, 08 Aug 1996 06:47:49 GMT

I am a web site far out there
```

Next time someone wants to access *http://192.168.124.1/message*, the proxy server does not have to lug bytes over the Web; it can just go and look it up.

There are a number of housekeeping directives that help with caching.

9.2.1 CacheRoot

```
CacheRoot directory
Default: none
Server config, virtual host
```

Sets the directory to contain cache files - must be writable by Apache.

9.2.2 CacheSize

```
CacheSize size_in_kilobytes
Default: 5
Server config, virtual host
```

This directive sets the size of the cache area in kilobytes. More may be stored, but garbage collection reduces it to less than the set number.

9.2.3 CacheGcInterval

```
CacheGcInterval hours
Default: never
Server config, virtual host
```

This directive specifies how often, in hours, Apache checks the cache and does a garbage collection if the amount of data exceeds `CacheSize`.

9.2.4 CacheMaxExpire

```
CacheMaxExpire hours
Default: 24
Server config, virtual host
```

This directive specifies how long cached documents are retained. This limit is enforced even if a document is supplied with an expiration date that is further in the future.

9.2.5 CacheLastModifiedFactor

```
CacheLastModifiedFactor factor
Default: 0.1
Server config, virtual host
```

If no expiration time is supplied with the document, then estimate one by multiplying the time since last modification by *factor*. `CacheMaxExpire` takes precedence.

9.2.6 CacheDefaultExpire

```
CacheDefaultExpire hours
Default: 1
Server config, virtual host
```

If the document is fetched by a protocol that does not support expiration times, use this number. `CacheMaxExpire` does not override it.

9.2.7 CacheDirLevels and CacheDirLength

```
CacheDirLevels number
Default: 3
CacheDirLength number
Default: 1
Server config, virtual host
```

The proxy module stores its cache with filenames that are a hash of the URL. The filename is split into `CacheDirLevels` of directory using `CacheDirLength` characters for each level. This is for efficiency when retrieving the files (a flat structure is very slow on most systems). So, for example:

```
CacheDirLevels 3
CacheDirLength 2
```

converts the hash "abcdefghijk" into *ab/cd/ef/ghijk*. A real hash is actually 22 characters long, each character being one of a possible 64 (2^6), so that three levels, each with a length of 1, gives 2^{18} directories. This number should be tuned to the anticipated number of cache entries (2^{18} being roughly a quarter million, and therefore good for caches up to several million entries in size).

9.2.8 CacheNegotiatedDocs

```
CacheNegotiatedDocs
Default: none
Server config, virtual host
```

If present in the Config file, this directive allows content-negotiated documents to be cached by proxy servers. This could mean that clients behind those proxys could retrieve versions of the documents that are not the best match for their abilities, but it will make caching more efficient.

This directive only applies to requests that come from HTTP/1.0 browsers. HTTP/1.1 provides much better control over the caching of negotiated documents, and this directive has no effect on responses to HTTP/1.1 requests.

9.2.9 NoCache

```
NoCache [host|domain] [host|domain] ...
```

This directive specifies a list of hosts and/or domains, separated by spaces, from which documents are not cached.

9.3 Setup

The cache directory for the proxy server has to be set up rather carefully with owner *webuser* and group *webgroup*, since it will be accessed by that insignificant person (see Chapter 2).

You now have to tell Netscape that you are going to be accessing the Web via a proxy. Click on Edit ➔ Preferences ➔ Advanced ➔ Proxies tab ➔ Manual Proxy Configuration. Click on View and, in the HTTP box, enter the IP address of our proxy, which is on the same network, 192.168.123, as our copy of Netscape:

```
192.168.123.4
```

Enter `8000` in the Port box.

For Microsoft Internet Explorer, select View ➔ Options ➔ Connection tab, check the Proxy Server checkbox, then click the Settings button and set up the HTTP proxy as described previously. That is all there is to setting up a real proxy server.

You might want to set up a simulation in order to watch it in action, as we did, before you do the real thing. However, it is not that easy to simulate a proxy server on one desktop, and when we have simulated it, the elements play different roles from those they have supported in demonstrations so far. We end up with four elements:

- Netscape running on a Windows 95 machine. Normally this is a person out there on the Web trying to get at our sales site; now, it simulates a Butterthlies member trying to get out.
- An imaginary firewall.
- A copy of Apache (site: *... /site.proxy/proxy*) running on the FreeBSD machine as proxy server to the Butterthlies site.
- Another copy of Apache, also running on FreeBSD (site: *... /site.proxy/real*) that simulates another web site "out there" that we are trying to access. We have to imagine that the illimitable wastes of the Web separate it from us.

The configuration in *... /site.proxy/proxy* is as shown earlier. Since the proxy server is running on a machine notionally on the other side of the Web from the machine running *... /site.proxy/real*, we need to put it on another port, usually 8000.

The configuration file in *... /proxy/real* is:

```
User webuser
Group webgroup
ServerName www.faraway.com

Listen www.faraway.com:80
DocumentRoot /usr/www/site.proxy/real/htdocs
```

On this site, we use the more compendious `Listen` with server name and port number combined. In *... /site.proxy/real/htdocs* there is a file message:

```
I am a web site far, far out there.
```

Also in */etc/hosts* there is an entry:

```
192.168.124.1 www.faraway.com
```

simulating a proper DNS registration for this far-off site. Note that it is on a different network (192.168.124) from the one we normally use (192.168.123), so that when we try to access it over our LAN, we can't without help. So much for *faraway*.

The weakness of all this is in */usr/www/lan_setup* on the FreeBSD machine, because we are trying to run these two servers, notionally on different parts of the Web, on the same machine:

```
ifconfig ep0 192.168.123.2
ifconfig ep0 192.168.123.3 alias netmask 0xFFFFFFFF
ifconfig ep0 192.168.124.1 alias
```

The script *lan_setup* has to map all three servers onto the same physical interface, *ep0*. The driver for *ep0* receives any request for these three IP numbers and forwards it to any copy of Apache via TCP/IP. Each copy of Apache tries to see if it has a virtual server with the number (and if it has, it handles the request), so we could find this setup appearing to work when really it isn't working.

Now for action: Get to Console 1 by pressing ALT-F1, go to *... /site.proxy/real*, and start the server with `./go`. Similarly, go to Console 2 and site *... /site.proxy/proxy*, and start it with `./go`. On Netscape, access *http://192.168.124.1/*.

You should see the following:

```
Index of /
. Parent Directory
. message
```

And if we select *message* we see:

```
I am a web site far out there
```

Fine, but are we fooling ourselves? Go to Netscape's Proxies page and disable the HTTP proxy by removing the IP address:

```
192.168.123.2
```

Exit from Netscape and reload; then reaccess *http://192.168.124.1/*. You should get some sort of network error.

What happened? We asked Netscape to retrieve *http://192.168.124.1/*. Since it is on network 192.168.123, it failed to find this address. So instead it used the proxy server at port 8000 on 192.168.123.2. It sent its message there:

```
GET http://192.168.124.1/ HTTP/1.0
```

The copy of Apache running on the FreeBSD machine, listening to port 8000, was offered this morsel and accepted the message. Since that copy of Apache had been told to service proxy requests, it retransmitted the request to the destination we thought it was bound for all the time, 192.168.123.1 (which it *can* do since it is on the same machine):

```
GET / HTTP/1.0
```

In real life, things are simpler: you only have to carry out steps 2 and 3, and you can ignore the theology. When you have finished with all this, remember to remove the HTTP proxy IP address from your browser setup.